Measurement Uncertainty in Artificial Intelligence

Valter Laino

UNIVERSITY OF SALERNO

DEPARTMENT OF INDUSTRIAL ENGINEERING

Ph.D. Course in Innovative Engineering Technologies for Industrial Sustainability
Curriculum in Electronic Engineering - XXXVIII Cycle

Measurement Uncertainty in Artificial Intelligence

Supervisor
Prof. Marco Carratù

Ph.D. Student
Valter Laino

Ph.D. Course Coordinator
Prof. Diego Barletta

Publications resulting from this work

Carratù, M., Gallo, V., Laino, V., & Liguori, C. (2023). Use of Artificial Intelligence in optical microscope imaging. In 19th IMEKO TC10 Conference-"MACRO meets NANO in Measurement for Diagnostics, Optimization and Control.

Carratù, M., Gallo, V., Laino, V., Liguori, C., Paciello, V., & Pietrosanto, A. (2023, October). The evaluation of uncertainty in measurements using Artificial Neural Network techniques. In IECON 2023-49th Annual Conference of the IEEE Industrial Electronics Society (pp. 1-4). IEEE.

Carratù, M., Gallo, V., Laino, V., Liguori, C., & Pietrosanto, A. (2023, October). A survey on Uncertainty Assessment in ANN-based Measurements. In 2023 IEEE International Conference on Metrology for eXtended Reality, Artificial Intelligence and Neural Engineering (MetroXRAINE) (pp. 1128-1133). IEEE.

Gallo, V., Shallari, I., Carratù, M., Laino, V., & Liguori, C. (2024). Design and characterization of a powered wheelchair autonomous guidance system. Sensors, 24(5), 1581.

Carratù, M., Gallo, V., Laino, V., Liguori, C., Pietrosanto, A., & Lundgren, J. (2024, May). Cross-Correlation Estimation in Artificial Neural Network for Uncertainty Assessment. In 2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) (pp. 1-6). IEEE.

Carratù, M., Gallo, V., Laino, V., Liguori, C., & Pietrosanto, A. (2024, May). Including Measurement Uncertainty to Improve the Reliability of Classification ANN. In 2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) (pp. 1-6). IEEE.

Lundgren, J., Jiang, M., Laino, V., Gallo, V., Carratù, M., & Nnonyelu, C. J. (2024, May). Accuracy Impact of Increased

Measurement Quality when using Pretrained Networks for Classification. In 2024 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) (pp. 1-5). IEEE.

Buonocore, D., Carratù, M., Gallo, V., Laino, V., Pietrosanto, A., & Sommella, P. (2024, May). Propagation of Measurement Uncertainty in IMU Orientation Tracking Algorithms. In 2024 IEEE International Workshop on Metrology for Industry 4.0 & IoT (MetroInd4. 0 & IoT) (pp. 65-69). IEEE.

Carratù, M., Gallo, V., Laino, V., Paciello, V., Espírito-Santo, A., & Shallari, I. (2024, July). Earthquake Magnitude Estimation with Single Seismic Station using Deep Learning. In 2024 IEEE International Symposium on Measurements & Networking (M&N) (pp. 1-6). IEEE.

Carratu, M., Laino, V., Pietrosanto, A., & Sommella, P. (2024, August). Improved Classification of Motorcycle Rear Stroke Suspension Sensor Faults. In 2024 IEEE 22nd International Conference on Industrial Informatics (INDIN) (pp. 1-14). IEEE.

Carratù, M., Di Leo, G., Gallo, V., Laino, V., Liguori, C., Paciello, V., & Pietrosanto, A. (2025). Reliability analysis of algorithms on a desk Quantum Computer. Measurement: Sensors, 101859.

Carratù, M., Gallo, V., Laino, V., Liguori, C., Pietrosanto, A., & Lundgren, J. (2025, May). Uncertainty-Aware Data Reconstruction in Autoencoders. In 2025 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) (pp. 1-6). IEEE.

Nnonyelu, C. J., Jiang, M., Gallo, V., Laino, V., Carratù, M., & Lundgren, J. (2025, May). Signal First-Difference as Augmentation Method for CNN-Based Heart Sound Classification. In 2025 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) (pp. 1-6). IEEE.

Carratù, M., Gallo, V., Laino, V., Sommella, P., Pietrosanto, A., Ciani, L., ... & Patrizi, G. (2025, May). Impact of Accelerated Life Testing on the Metrological Performance of Commercial MEMS Accelerometers. In 2025 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) (pp. 1-6). IEEE.

Carratù, M., Gallo, V., Laino, V., Sommella, P., Ciani, L., & Patrizi, G. (2025). Reliability Assessment of MEMS IMUs based on Uncertainty Propagation in Positioning Algorithms during Accelerated Life Test. IEEE Transactions on Instrumentation and Measurement, 1–1. https://doi.org/10.1109/TIM.2025.3619220

Shallari, I., Gallo, V., Laino, V., & Carratú, M. (2025). Robust Distance Estimation Using LiDAR-Guided Deep Learning for Assistive Mobility. IEEE Transactions on Instrumentation and Measurement, 74, 1–15. https://doi.org/10.1109/TIM.2025.3637978

Carratù, M., Gallo, V., Laino, V., Liguori, C., & Pietrosanto, A. (2025). Input Data Measurement Uncertainty Propagation in Artificial Neural Networks. IEEE Open Journal of Instrumentation and Measurement, 4, 1–12. https://doi.org/10.1109/OJIM.2025.3643040

Contents

List of figures

List of tables

Abstract

The rapid proliferation of Artificial Intelligence (AI) systems for classification and prediction has transformed both scientific research and industrial applications. While these technologies have demonstrated remarkable performance, their deployment in safety-critical contexts raises fundamental concerns regarding the reliability, robustness, and trustworthiness of their outputs. In particular, inaccurate estimations may result in high-risk or even life-threatening consequences, underscoring the need for rigorous methods to quantify and assess the uncertainty associated with AI-driven estimates.

This project addresses these challenges by introducing a novel methodology grounded in the Law of Propagation of Uncertainty (LPU). The approach is designed to provide fast and simultaneous inference of measurement uncertainty in Artificial Neural Networks (ANNs), thereby enabling systematic evaluation of the effects of uncertainty propagation from input data to final outputs. Unlike conventional black-box treatments of neural networks, the proposed methodology seeks to explicitly analyze their underlying mathematical structures and account for the nonlinearities that complicate direct application of standard uncertainty analysis. By incorporating LPU into the inference process, this work enables the propagation of input measurement uncertainty through the network architecture, yielding an analytical estimation of output uncertainty. This provides a rigorous framework aligned with ISO-GUM. The ultimate goal is to enhance the interpretability, transparency, and reliability of ANN-based systems, particularly in domains where decision-making depends critically on the validity of predictions.

Introduction

Artificial Neural Networks (ANNs) have become indispensable tools in modern data-driven applications, spanning healthcare diagnostics, financial forecasting, autonomous systems, industrial monitoring, and cybersecurity. Their strength lies in their ability to capture complex nonlinear relationships from large, high-dimensional datasets, making them highly effective predictive models (Basheer & Hajmeer, 2000). However, in safety-critical and high-stakes domains, prediction accuracy alone is insufficient: incorrect or overconfident outputs can lead to severe consequences, including risks to human safety, substantial financial losses, or catastrophic system failures (Hramov et al., 2018; Kabir et al., 2018). Despite their widespread adoption, ANNs remain fundamentally limited in transparency and interpretability. Often treated as black-box models, they provide predictions without explicit measures of reliability, which is problematic in contexts that demand compliance with rigorous standards, such as metrology, where precision, repeatability, and traceability are essential. To ensure that ANN outputs can be trusted in such domains, it is critical to complement predictions with measures that rigorously quantify their reliability (Kasiviswanathan et al., 2016).

Uncertainty Quantification (UQ) provides a principled framework to express the confidence associated with individual predictions (Soize, 2017; Tripathy & Bilionis, 2018). Unlike traditional performance metrics (e.g., accuracy or mean squared error), which offer only global assessments of model performance, UQ allows users to distinguish between predictions that are trustworthy and those that require further scrutiny, additional data, or expert evaluation (Wang et al., 2025). When aligned with metrological principles, ANN-based UQ enables neural networks to function as “soft sensors” or virtual instruments whose performance can be characterized, validated, and trusted (Shirmohammadi et al., 2024). Such alignment supports the adoption of AI systems in domains that require accountability in addition to predictive power (Abdar et al., 2021).

Recent work in instrumentation and measurement further emphasizes that UQ for data-driven AI-assisted measurements should be treated as an integral component of the measurement process rather than as an optional add-on to prediction. In particular, Shirmohammadi et al. (2025) propose a holistic viewpoint bridging measurement science and AI practice, including a taxonomy of UQ approaches and a discussion of recurring practices in the AI literature that are noncompliant with measurement standards. For classification-assisted measurements, they also stress the importance of evaluating UQ using measurement-oriented criteria, such as the impact on

indirect measurement accuracy, alongside conventional classification metrics (precision, recall, and F1 score in both macro and weighted modes).

In line with this measurement-oriented perspective, this thesis treats the ANN as an indirect measurement model (i.e., a soft sensor) whose outputs must be accompanied by uncertainty information that is traceable and compatible with established metrological practice. While many current uncertainty estimation approaches, such as Monte Carlo simulations, ensemble methods, and Bayesian neural networks, can provide useful uncertainty indicators, they often require substantial computational resources and can be difficult to scale for real-time or resource-constrained settings; additionally, they may not fully comply with established metrological standards in precision-critical contexts.

To address these limitations, this work introduces an analytical framework based on the law of propagation of uncertainty (LPU) for both regression and classification ANNs, and for a range of commonly used deterministic architectures and differentiable activation functions. By propagating input measurement uncertainties through a deterministic ANN, the method provides fast, closed form estimates of the output's aleatory uncertainty without repeated stochastic sampling or repeated model evaluations, making it suitable for real-time environments. Crucially, the framework is compliant with ISO-GUM (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML. Evaluation of measurement data — Guide to the expression of uncertainty in measurement. Joint Committee for Guides in Metrology, JCGM 100:2008. doi:10.59161/JCGM100-2008E), ensuring consistency with established practices in metrology and measurement science. The confidence level reported for a class assignment is obtained by combining the ANN output with the propagated uncertainty through probabilistic inference; it should not be conflated with raw misclassification probability or a softmax score, which are not uncertainty per se in a metrological sense.

Overall, the main contribution of this work is the systematic coupling of ANN predictions with rigorously quantified confidence levels, enhancing interpretability, transparency, and reliability. This is particularly relevant in domains where decision-making cannot rely solely on accuracy but must also consider the model's confidence, including healthcare, autonomous systems, industrial automation, and scientific measurement. By integrating metrological rigor into ANN-based systems, this framework advances the trustworthiness and practical deployment of neural networks in environments where uncertainty is as critical as prediction accuracy.

This thesis is driven by a set of research questions aimed at making uncertainty propagation in neural networks both metrologically meaningful and practically usable. The first question concerns ISO-GUM consistency: how a deterministic ANN can be framed as an indirect measurement model (soft sensor) whose outputs are accompanied by uncertainty information expressed in ISO-GUM terms. The second question addresses analytical

propagation: how prescribed input measurement uncertainty can be propagated through common deterministic ANN architectures and differentiable activation functions while preserving the nominal deterministic prediction at the nominal input. The third question focuses on validity: under which conditions local linear propagation provides reliable uncertainty estimates, and how its behaviour relates to Monte Carlo propagation when the network is nonlinear over the operating region. The fourth question concerns decision making: how propagated uncertainty can be used in binary and multi-class classification to produce confidence-driven decisions, and how such confidence should be interpreted in a metrological sense rather than being conflated with softmax scores or misclassification probabilities. The fifth question extends the discussion to reconstruction problems: in unsupervised dimensionality reduction and signal reconstruction, what propagated uncertainty represents and how uncertainty bands should be interpreted when reconstruction/model error contributes to the residual.

Accordingly, the objectives of this thesis are to formulate an ISO-GUM-aligned analytical propagation framework based on the law of propagation of uncertainty and local sensitivities; to provide practical propagation rules and implementation guidance for common architectures, activation functions, and preprocessing/normalization steps so that uncertainties remain consistent in the network input space; to clarify the conceptual and computational differences between LPU-based propagation and Monte Carlo propagation including the distinction between the deterministic prediction and the expectation; to demonstrate how propagated uncertainty can be incorporated into classification decisions in both binary and multi-class settings; and to illustrate uncertainty propagation in reconstruction models while discussing interpretation limits and practical implications.

The results of this thesis have been disseminated through several peer-reviewed publications developed with co-authors. To clarify how these works relate to the present manuscript, the thesis explicitly maps each publication to the corresponding chapters/sections and briefly summarizes the PhD candidate's concrete contributions (e.g., methodological development, implementation, experimental design, analysis, and writing), as well as the elements that are consolidated or extended beyond the published material.

Finally, it is worth noting that the proposed LPU-based uncertainty propagation relies on a local linearization of the ANN mapping around the current operating point; therefore, its accuracy depends on both the network's local nonlinearity (e.g., saturation regimes) and the magnitude of the input uncertainties. For this reason, the thesis explicitly delineates the boundary conditions under which the analytical estimates are expected to remain valid: the method applies directly to deterministic architectures whose input-output mapping is differentiable with respect to the inputs (including common feedforward and convolutional models with standard differentiable activation functions), while architectures involving non-differentiable operations or

recurrent dynamics may require additional treatment (e.g., smooth approximations or time-unrolling) and are not the primary scope of the present validation. Practical operative guidelines are also provided to support application in real settings, including recommendations on preprocessing/normalization, monitoring activation operating regions, and using limited Monte Carlo checks to verify that the selected operating range remains within the linear regime assumed by the propagation law.

Chapter I
Uncertainty Assessment in AI

I.1 Artificial Intelligence

Artificial Intelligence can be defined as the ability of a system to emulate certain cognitive functions typically associated with the human mind, such as reasoning, learning, planning, and creativity. In other words, the goal is to design machines capable of replicating human thought processes, enabling computers to perform complex tasks and address practical problems across a wide variety of domains (Jiang et al., 2022). AI is built upon a combination of technologies and methodologies that originate from different disciplines, including computer science, psychology, linguistics, mathematics, and even neuroscience, making it a deeply interdisciplinary field (Han, 2026). This convergence of knowledge allows AI not only to simulate isolated aspects of intelligence but also to integrate them in ways that approach human-like adaptability and problem-solving. At its core, AI can be seen as comprising two major components: Machine Learning (ML) (Alpaydin, 2021) and Deep Learning (DL) (Goodfellow et al., 2016) (Figure I.1). Machine Learning focuses on algorithms that enable systems to learn patterns and improve their performance from data without being explicitly programmed. DL, a subset of ML, leverages ANNs with many layers to model complex representations and achieve state-of-the-art results in areas such as computer vision, natural language processing, and speech recognition. Together, ML and DL provide the foundation for much of the progress in AI, empowering machines not only to analyze and interpret information but also to make decisions, adapt to new circumstances, and generate novel solutions (Kufel et al., 2023).

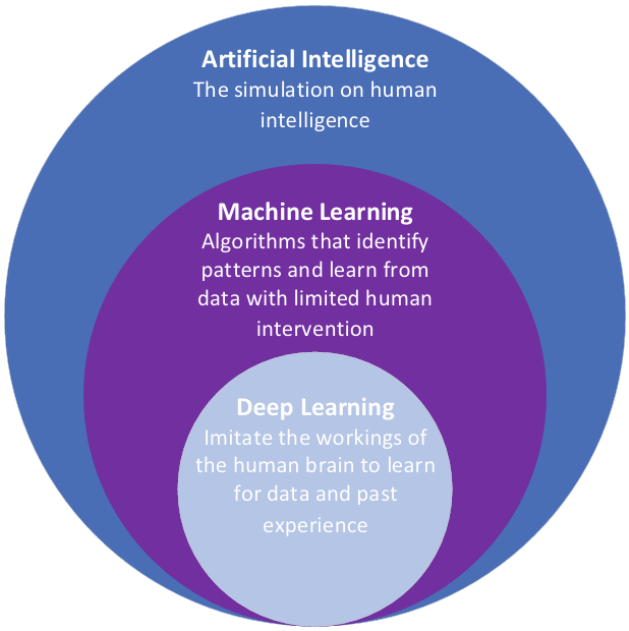

Figure I.1 *AI ecosystem (Fergus & Chalmers, 2022).*

I.1.1 Machine Learning

Machine Learning constitutes a subfield of AI dedicated to the development of systems capable of acquiring knowledge and improving their performance through the analysis of data (Zhou, 2021). In this framework, machines can execute specific tasks without being explicitly programmed to do so (Simon, 2013). Presently, ML is employed across numerous domains, including natural language processing, computer vision, cognitive computing, and knowledge representation. The operational principles of ML can be analogized to human cognitive development. A newborn child, initially incapable of self-sufficiency, progressively attains autonomy and the ability to perform increasingly complex tasks through experience. For instance, a young child is initially unable to distinguish between an apple and a banana; this ability emerges through exposure and the accumulation of experience, which fosters the recognition of characteristic attributes such as color, shape, and texture.

Similarly, an ML system is designed to infer patterns from data, enabling it to recognize objects and make decisions. For example, the system may learn that apples are generally round while bananas are elongated and yellow. Such characteristics are mathematically encoded within a dataset, facilitating systematic analysis and processing.

A dataset comprises a set of variables, termed features, which encapsulate the observable attributes of each object. The complete set of features associated with a single data instance is referred to as a feature vector. While datasets contain substantial volumes of information, the judicious selection of features is essential to mitigate redundancy, instances in which multiple features convey overlapping information. The principal objective of an ML system is to discern patterns within these features and construct a predictive function capable of accurately generalizing to new, unseen inputs (Paullada et al., 2021). The role of the practitioner is, therefore, to ensure that the dataset is appropriately curated and that the system is guided to effectively leverage the knowledge it acquires. In measurement-driven applications, these features often correspond to sensor-derived quantities affected by measurement uncertainty; quantifying how such input uncertainty propagates through a trained model is a central objective of this thesis.

During the learning process, particularly in its initial stages, errors are to be expected. Nevertheless, as the system is exposed to progressively larger quantities of data, its predictive accuracy typically improves. This underscores the necessity of sufficiently extensive datasets to enhance the system's capacity for generalization. Conversely, excessively large or unrefined datasets may impede the algorithm's efficacy by introducing superfluous complexity or noise, thereby complicating the extraction of meaningful patterns (Verbraeken et al., 2020).

In essence, ML obviates the need for manual programming of knowledge. Instead, it derives insights from empirical examples and experiential data. Rather than developing an explicit program to address a specific problem, practitioners provide numerous instances that illustrate the correct output for given inputs. The resultant model, constituted by a multitude of numerical parameters, is subsequently capable of autonomous interpretation, adaptation, and refinement, thereby continually enhancing its performance over time.

In Machine Learning, different learning paradigms exist depending on how information is provided during training. In this thesis, the focus is on paradigms that are directly relevant to uncertainty propagation in deterministic neural networks, namely supervised learning and, when pre-trained models are adopted, transfer learning.

The paradigms most relevant to this dissertation are:

- Supervised learning (main focus): the model is trained on labeled input-output pairs to learn a mapping used for regression or classification. This setting matches the use cases investigated in this work, where the trained ANN is treated as an indirect measurement model and input features may carry measurement uncertainty.
- Transfer learning (relevant in practice): knowledge from a source task/domain is reused to improve learning in a related target task,

typically by starting from a pre-trained network and adapting it. This is relevant because the proposed uncertainty-propagation approach applies to a fixed, deterministic ANN mapping, regardless of whether it is trained from scratch or obtained via transfer learning.

Other paradigms (unsupervised, semi-supervised, and reinforcement learning) are widely used in representation learning, data-scarce scenarios, and sequential decision-making, respectively, but they are not central to the scope of the present thesis.

Machine learning includes several model families (e.g., linear models, tree-based methods, kernel methods, and neural networks), and the choice of model influences both performance and interpretability. Since the contributions of this dissertation rely on ISO-GUM-compliant analytical uncertainty propagation, the focus is placed on deterministic and differentiable input-output mappings for which sensitivity information can be derived. For this reason, the following section concentrates on Artificial Neural Networks (ANNs).

I.1.2 Artificial Neural Networks

An Artificial Neural Network is a computational model designed to make decisions in a manner that resembles the functioning of the human brain, by replicating the information-processing mechanisms typical of biological neurons (Yegnanarayana, 2009). The principal objective of neural networks is to emulate, through software or hardware implementations, the processes of learning and information transmission that occur in the brain via synaptic connections.

In essence, an ANN is composed of interconnected units, or artificial neurons, organized in layers. Each neuron receives input signals, processes them through an activation function, and transmits the resulting output to subsequent neurons. This mechanism mirrors the way biological neurons transmit signals through synapses, allowing the network to propagate and transform information across multiple layers (Islam et al., 2019).

Most ANNs are adaptive systems, meaning that they adjust their internal parameters, commonly referred to as weights and biases, based on the data they are exposed to during the training phase. Through iterative optimization processes such as backpropagation and gradient descent, the network progressively refines its structure, thereby improving its performance on specific tasks. Neural networks constitute a cornerstone of ML, as their architecture enables the automatic discovery of complex and non-linear relationships within data. Their learning capability is directly inspired by the behavior of biological neurons: just as a human brain strengthens or weakens synaptic connections based on experience, an ANN learns by adjusting the strength of connections between artificial neurons in response to training data.

Thanks to their flexibility, neural networks underpin a wide range of modern applications, including image and speech recognition, natural language processing, autonomous systems, and predictive modeling. They have become the foundation of DL, where multiple layers of neurons are stacked to create highly expressive models capable of extracting hierarchical features and achieving state-of-the-art results in numerous domains (Wu & Feng, 2018).

I.1.2.1 ANNs History

The history of ANNs is marked by a complex trajectory, characterized by alternating phases of groundbreaking innovation and periods of stagnation, which have collectively shaped the networks as we know them today. The fundamental idea, as previously mentioned, was to imitate the functioning of the human brain. In the early 1940s, Warren McCulloch (a psychiatrist and neuroanatomist) and Walter Pitts (a mathematician) proposed the first mathematical model of an artificial neuron (McCulloch & Pitts, 1943). Subsequently, psychologist Donald Hebb introduced a learning theory that is now regarded as a prototypical example of Unsupervised Learning (Hebb, 1949).

In the 1950s, Frank Rosenblatt developed the Perceptron (Rosenblatt, 1958), an early learning algorithm and the first practical implementation of an ANN capable of solving simple binary classification problems.

After the initial surge of research in the 1950s and 1960s, the field entered its first major "AI winter," a prolonged decline in interest that lasted nearly two decades (Toosi et al., 2021). A significant factor contributing to this decline was the publication of Perceptrons by Marvin Minsky and Seymour Papert (Minsky & Papert, 1969), which rigorously highlighted the limitations of single-layer perceptrons. Attention then shifted toward exploring alternative network architectures capable of overcoming these shortcomings.

The introduction of Multi-Layer Perceptrons represented a turning point. By connecting multiple layers of perceptrons, researchers developed the first feedforward multilayer networks, enabling the representation of more complex, non-linear relationships (Kruse et al., 2022). Crucially, the invention and adoption of the backpropagation algorithm (error back-propagation) provided a practical means to train such networks. This advancement, together with increasing computational resources, sparked a renewed wave of interest in neural networks during the 1980s. The same period also saw the conceptual development of RNNs and CNNs, architectures that would later become central to deep learning.

In the 1990s, however, the study of neural networks once again experienced a decline. The rise of alternative ML techniques, most notably Support Vector Machines, offered superior performance in many applications at the time. As a result, neural networks temporarily fell into relative obscurity.

The dawn of the 21st century brought about a decisive breakthrough, driven by three converging factors: the availability of massive datasets, advances in computational power through specialized hardware (such as GPUs), and novel approaches to training deep architectures (Jeon et al., 2021). A pivotal figure in this revival was Geoffrey Hinton, whose work demonstrated the effective training of Deep Neural Networks (DNNs) and reignited global interest in the field (Hinton et al., 2012). From that moment, deep architectures such as CNNs and RNNs rapidly became the state of the art, achieving remarkable success in diverse domains, including speech recognition, automatic translation, computer vision, and content generation (Yin et al., 2017).

Today, ANNs constitute one of the foundational technologies of modern AI. They underpin many of the most significant innovations of the last two decades and remain an indispensable tool for tackling increasingly complex and multidisciplinary challenges.

I.1.2.2 ANNs Operation

To understand how ANNs function, it is first necessary to explore the fundamentals of how the human brain operates (Sharma et al., 2017). The nervous system, which serves as the biological foundation of cognition and response, can be conceptually divided into three main stages:

- The neural network: continuously receives information from the environment and processes it to make appropriate decisions.
- The receptors: convert external stimuli (such as light, sound, or pressure) into electrical impulses, which are then transmitted to the neural network.
- The effectors (actuators): transform the electrical signals generated by the neural network into physical or chemical responses that can act on the external system (for example, muscle contractions or glandular secretions).

The neural network itself is composed of numerous interconnected neurons. A biological neuron is a specialized cell characterized by three essential structural components:

- The cell body (soma): which contains the nucleus, responsible for directing all cellular activities. The soma is also the point from which dendrites branch out.
- The axon: a single long fiber that transmits electrical signals from the cell body to the dendrites of other neurons.
- The dendrites: branching structures that receive signals from other neurons and convey them to the soma, where the information is integrated.

Neurons operate by receiving electrical inputs through their dendrites. When an incoming impulse reaches the soma, it produces a change in the cell's

electrical potential. The soma functions somewhat like a capacitor, accumulating electrical energy. However, this storage capacity is limited. Once the accumulated energy exceeds a critical threshold, the neuron is activated and generates an action potential, which propagates along the axon (Montesinos López et al., 2022).

When the action potential reaches the terminal end of the axon, it triggers communication at the synapse, the junction between the axon of one neuron and the dendrites of another. At this point, the electrical signal is converted into a chemical signal through the release of neurotransmitters. These chemical messengers cross the synaptic gap and bind to receptors on the receiving neuron, influencing its likelihood of activation. Depending on the nature of the neurotransmitter and the receptor, the synapse may be excitatory (increasing the chance of neuron activation) or inhibitory (reducing the chance of activation).

Furthermore, neurons are not capable of firing continuously without interruption. After transmitting an impulse, they enter a short refractory period during which they cannot immediately generate another action potential. This temporal limitation ensures controlled and regulated neural communication.

In analogy with biological neural networks, an Artificial Neural Network is composed of a collection of interconnected computational units known as artificial neurons (Sharma & Dev, 2012) (Figure I.2). These units are specifically designed to extract meaningful patterns from data and to store the acquired knowledge through adjustable parameters called synaptic weights.

Thanks to their layered architecture and the use of activation functions, which may be either linear or nonlinear, ANNs are theoretically capable of approximating any mathematical function to an arbitrary degree of accuracy. This property, known as the universal approximation theorem, explains why ANNs can be successfully applied to a wide range of problems.

In a linear problem, such as calculating the total cost of a certain number of apples when the price per apple is fixed, the relationship can be expressed by a simple linear equation. In contrast, predicting the market price of a house is far more complex. The price depends on multiple factors, such as the size of the building, the presence of a garden, the quality of the neighborhood, or proximity to essential services. The interaction of these variables introduces nonlinear relationships, making the problem much harder to solve with traditional linear models. ANNs, however, can learn and represent such nonlinear dependencies with remarkable effectiveness.

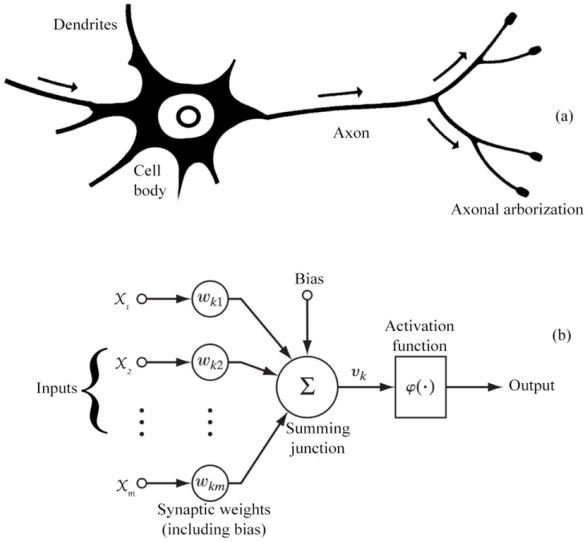

Figure I.2 *Neuron Modeling (Arbib et al., 2003, Haykin, 2009).*

I.1.2.3 Artificial Neuron

An artificial neuron is designed to receive, process, and transmit information in the form of numerical signals. In its most basic form, it can be described through three essential components (Kukreja et al., 2016):

- Weights and bias: adjustable parameters that determine the relative importance of each input and introduce flexibility in the decision boundary.
- Summation operator: responsible for aggregating the weighted inputs into a single scalar value.
- Activation function: a nonlinear (or in some cases linear) transformation that determines the final output of the neuron.

The earliest mathematical abstraction of a neuron was proposed by McCulloch and Pitts (1943) and later refined by Rosenblatt (1958) with the introduction of the perceptron model. In this framework, the neuron receives as input a set of signals, which represent the features or characteristics of the data to be analyzed. Each input is associated with a corresponding weight, reflecting its relative importance in the decision-making process.

The neuron also includes a bias term, which can be interpreted as a threshold or offset. The bias allows the activation function to shift left or right, thereby enabling the model to represent a broader class of functions.

To better understand how this structure works, it is useful to consider a simple practical example. Imagine that the task of the system is to recognize a specific object within an image, for instance, a cat.

The inputs to the artificial neuron correspond to different features of the image: the presence of whiskers, the shape of the ears, or the characteristic fur pattern. Each input is represented by a numerical value that quantifies how strongly that feature is present in the given image. The weights associated with the inputs express the relative importance of each feature in determining whether the object is a cat. For example, if ear shape is generally a more reliable indicator than fur color, the system may assign a higher weight to that feature. The neuron then processes the information by computing a weighted sum of all the inputs, to which a bias is added. The bias serves to shift the decision threshold, giving the model more flexibility in how it classifies inputs. This aggregated value is subsequently passed through an activation function, which determines the neuron's final output. Depending on whether the result exceeds a certain threshold, the neuron will decide whether the object in the image is likely to be a cat or not. Of course, this is only a highly simplified description of the behavior of a single artificial neuron. In reality, modern ANNs consist of thousands or even millions of interconnected neurons, distributed across multiple layers, working together to extract increasingly complex features and to perform sophisticated tasks such as image recognition, natural language processing, or autonomous decision-making.

The activation function represents the most crucial component of an artificial neuron. It determines the neuron's output and is directly responsible for introducing nonlinearity into the model (Rasamoelina et al. 2020). Without activation functions, the entire neural network would collapse into nothing more than a linear combination of its inputs, regardless of the number of layers. This would severely limit its ability to solve complex problems, since many real-world relationships are inherently nonlinear (Rodríguez & Buitrago, 2022).

Activation functions play several essential roles:

- They enable the network to capture nonlinear patterns in data.
- They regulate the flow of information by deciding whether a neuron should be "activated" or remain silent.
- They contribute to the stability and efficiency of the learning process, often influencing convergence speed during training.

Over the years, various activation functions have been developed, each with specific advantages and drawbacks depending on the task. Among the most widely used are (Mercioni & Holban, 2020) (Figures I.3-I.7):

- Step function: one of the earliest activation functions, used in the original McCulloch–Pitts model. It outputs a binary value (0 or 1) depending on whether the input exceeds a threshold. While conceptually simple and biologically inspired, it lacks differentiability, making it unsuitable for modern training algorithms based on gradient descent. The mathematical expression defining the Step function is provided in Equation (I.1).

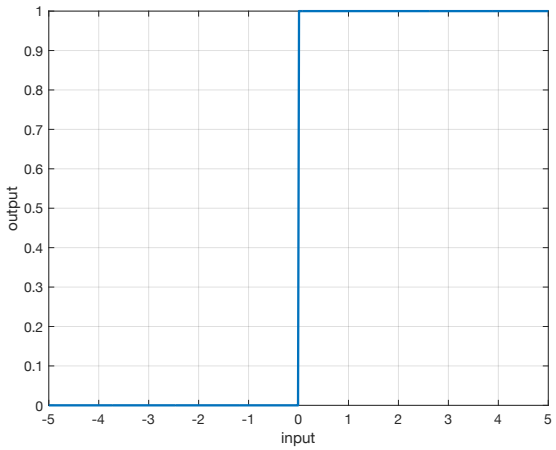

Figure I.3 *Step function.*

$$(\upsilon_k) = \begin{cases} 1, \ \upsilon_k > 0 \\ 0, \ \upsilon_k \leq 0 \end{cases} \qquad \text{(I.1)}$$

- Rectified Linear Unit (ReLU): currently the most popular choice. It introduces sparsity in activations and reduces computational complexity, though it may suffer from the "dying ReLU" problem. The mathematical expression defining the ReLU is provided in Equation (I.2).

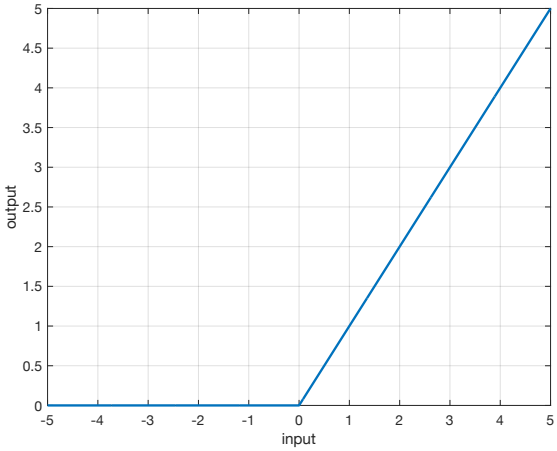

Figure I.4 *ReLU function.*

$$(v_k) = \begin{cases} v_k, \ v_k > 0 \\ 0, \ v_k \leq 0 \end{cases} \tag{I.2}$$

- Sigmoid function: maps inputs to a range between 0 and 1, making it useful for probabilistic interpretations, though it can suffer from vanishing gradients. The mathematical expression defining the Sigmoid function is provided in Equation (I.3).

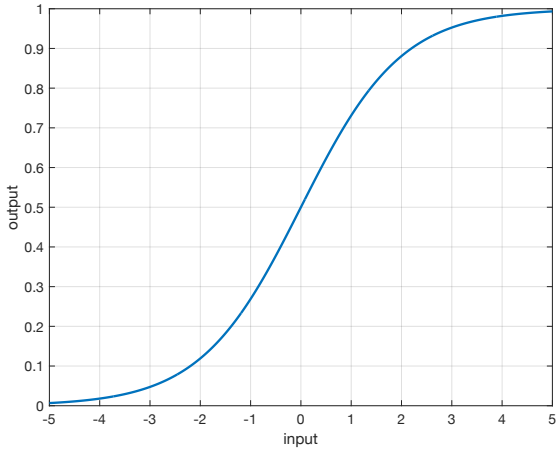

Figure I.5 *Sigmoid function.*

$$\varphi(v_k) = \frac{1}{1 + \ e^{-v_k}} \tag{I.3}$$

- Hyperbolic tangent (tanh): similar to the sigmoid but outputs values between –1 and 1, allowing for stronger gradients and a more balanced output. The mathematical expression defining the tanh is provided in Equation (I.4).

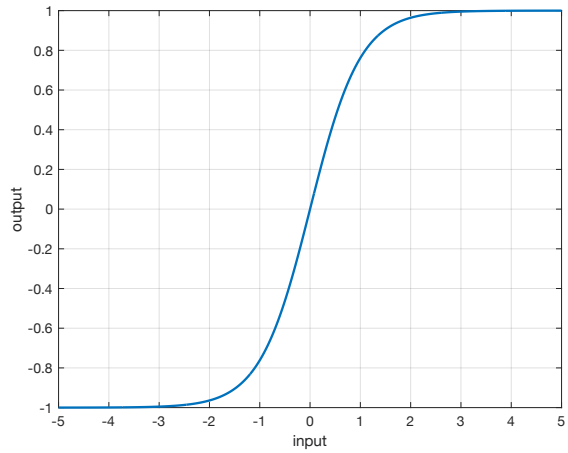

Figure I.6 *Hyperbolic Tangent function.*

$$\varphi(v_k) = \frac{e^{v_k} - e^{-v_k}}{e^{v_k} + e^{-v_k}} \tag{I.4}$$

- Softmax function: often used in the output layer of classification networks, as it converts raw scores into normalized probabilities across multiple classes. The mathematical expression defining the softmax function is provided in Equation (I.5).

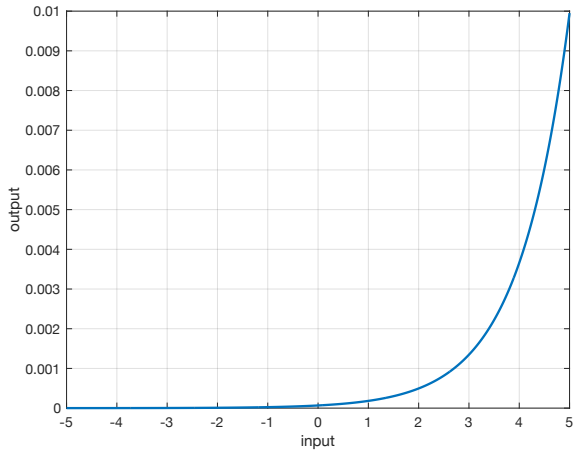

Figure I.7 *Softmax function.*

$$\varphi(v_{k_i}) = \frac{e^{v_{k_i}}}{\sum_{j=1}^{K} e^{v_{k_j}}} \quad \text{(I.5)}$$

In summary, activation functions are not only mathematical tools but also the key mechanism that gives ANNs their expressive power, allowing them to approximate complex nonlinear functions and tackle tasks such as image recognition, speech processing, and natural language understanding.

I.1.2.4 ANNs Structure

Inspired by the structure of biological neural networks, researchers have developed a simplified artificial system capable of processing information efficiently. As discussed in the previous sections, the fundamental building blocks of this system are artificial neurons, which are interconnected to form a graph-like structure.

The basic architecture of an ANN (Figure I.8) typically consists of three main types of layers (Qamar & Zardari, 2023):

- Input layer: This is the entry point of the network, where raw data are received. The input is usually represented as a fixed-length numerical vector, defined by the user according to the problem at hand. Each node in the input layer corresponds to a specific feature of the dataset (for example, pixel intensity in an image or a numerical attribute in a dataset).
- Hidden layers: Located between the input and the output, hidden layers are where most of the computation and feature extraction occur. The number of hidden layers, as well as the number of neurons within each layer, can vary depending on the complexity of the problem. These layers introduce nonlinearity into the model, allowing the network to capture intricate patterns and relationships that cannot be represented by linear mappings alone.
- Output layer: The final layer of the network, which generates the predicted result based on the computations performed by the preceding layers. The output is typically expressed as a fixed-length numerical vector, again defined by the user. The number of neurons in this layer corresponds to the number of possible outputs or classes (e.g., single neuron for binary classification, one per category in multi-class classification).

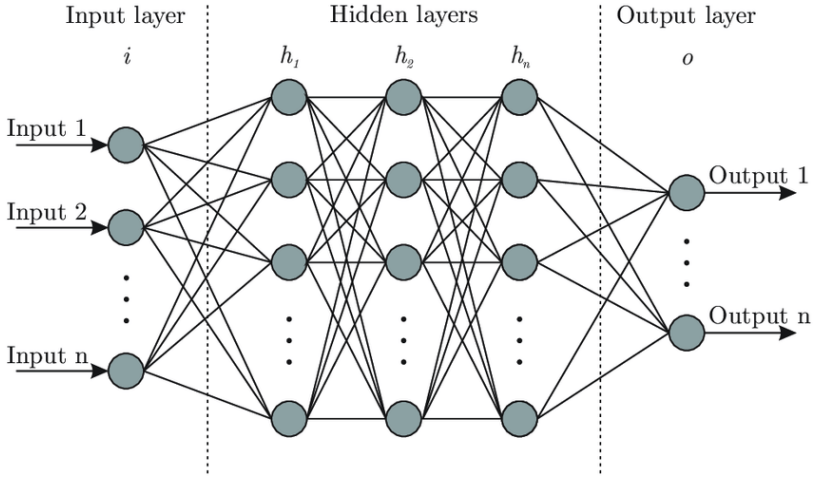

Figure I.8 *ANN structure (Bre et al., 2018).*

The computational flow within the network follows a feedforward process: the input values are transmitted to all the neurons in the first hidden layer, whose outputs then serve as inputs to the next layer, and so on. The final output layer produces the actual prediction or decision of the network.

While the number of input and output nodes is determined by the structure of the data and the nature of the task, the number of hidden nodes can be chosen more flexibly. Increasing the number of neurons or hidden layers typically enhances the network's ability to represent complex functions, but it also raises computational cost and the risk of overfitting.

A neural network with two or more hidden layers is referred to as a DNN, and the associated learning paradigm is known as Deep Learning. Deep architectures have proven particularly effective in solving highly complex problems, such as image and speech recognition, natural language understanding, and game-playing systems, where shallow models would fail to capture the richness of the data (Namatēvs, 2017).

Depending on the learning algorithm employed, it is often preferable to adopt a specific type of neural network architecture. In general, two broad classes of networks are commonly distinguished:

- Feedforward Neural Networks: In these architectures, information flows strictly in one direction-from the input layer, through one or more hidden layers, to the output layer-without any feedback loops (Bebis & Georgiopoulos, 2002). Feedforward networks can be further categorized into:
  - Single-layer networks, which consist only of an input layer directly connected to an output layer. These are relatively simple models, typically used for linearly separable problems.

  - Multilayer networks, which include one or more hidden layers between input and output. Thanks to nonlinear activation functions, they can approximate highly complex functions and solve nonlinearly separable problems.
- Recurrent Neural Networks (RNNs) (Medsker & Jain, 2001), also known as feedback networks: Unlike feedforward models, RNNs incorporate loops in their structure, allowing outputs from previous steps to be fed back as inputs in subsequent steps. This internal memory mechanism enables them to capture temporal dependencies and contextual information, making them particularly well-suited for sequential data such as time series, speech, and natural language.

These two classes represent the foundation of modern neural architectures. While FFNNs (Feedforward Neural Networks) are often employed in static pattern recognition and function approximation tasks, RNNs excel in domains where order and context matter, extending the expressive power of neural networks to dynamic and temporal data.

A more advanced class of neural networks is represented by Convolutional Neural Networks. CNNs belong to the family of multilayer feedforward networks, but they are distinguished by their highly specialized architecture. Typically, a CNN is composed of at least five layers, although modern implementations often contain dozens or even hundreds of layers in very deep models.

Within the hidden layers of a CNN, some are specifically designated as convolutional layers. These layers perform a mathematical operation known as convolution, which applies a set of learnable filters (or kernels) across the input. The result of this operation is the generation of a feature map, which highlights important local patterns such as edges, textures, or shapes (O'shea & Nash, 2015). Each feature map is essentially a transformation of the input received from the previous layer, and as the network goes deeper, these transformations allow the extraction of increasingly abstract and high-level features.

This hierarchical feature extraction process significantly enhances the accuracy of the network's predictions, making CNNs exceptionally powerful for analyzing high-dimensional and complex data, such as images, audio signals, or video streams (Li et al., 2021; O'Shea & Nash, 2015).

I.1.2.5 ANNs Training

The training of Artificial Neural Networks is a crucial yet highly complex process, as it involves adjusting the network's parameters to optimize system performance. This process requires modifying the weights associated with the

connections between artificial neurons to minimize the error between the network's output and the desired target value (Afaq & Rao, 2020).

Training requires the availability of a training dataset consisting of input-output pairs, which allows the network to learn the mapping between them. Consider a dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, where each input x_i is associated with a target output y_i. Initially, the network is initialized with random weights and biases. An input x_i is propagated through the network layer by layer until it reaches the output level (Forward Propagation). Each neuron processes the received data and applies an activation function to introduce nonlinearity.

The neural network implements a parametric mapping $f(\cdot;\theta)$. For each input x_i, the network produces the prediction $\hat{y}_i = f(x_i;\theta)$, which is compared with the corresponding target y_i through a loss function that quantifies their discrepancy (Janocha & Czarnecki, 2017).

For regression problems, the loss function is often the Mean Squared Error (MSE)**:**

$$L(D) = \frac{1}{N}\sum_{i=1}^{N}(f(x_i;\theta) - y_i)^2. \tag{I.6}$$

Ideally, this error term should be zero, meaning the network outputs perfectly match the training data. In practice, the goal is to make the error as small as possible. The computed error is then propagated backward through the network, from the output layer to the input layer (Backpropagation) (Rumelhart et al., 1986). Assuming the network uses differentiable activation functions, the error term itself is differentiable, making it possible to compute the gradient of the loss function with respect to the network parameters $\theta = \{w_1, b_1, \dots, w_N, b_N\}$, where w and b are weights and biases, respectively:

$$g = \nabla_\theta L. \tag{I.7}$$

During backpropagation, this gradient provides the direction in which the weights should be modified to reduce the error. The parameters are then updated using an optimization algorithm, with updates scaled by a learning rate ϵ_g :

$$\theta = \theta - \epsilon_g g. \tag{I.8}$$

The learning rate controls the magnitude of the weight updates at each iteration (Igiri et al., 2015). If too large, training may become unstable, while a too small value can lead to slow convergence.

This training process is repeated for a predefined number of iterations, called epochs, until convergence is achieved or a maximum number of epochs is reached. The number of epochs must be chosen carefully: too few prevent

the model from approaching the minimum of the loss function, while too many can cause overfitting (Afaq & Rao, 2020).

To better understand optimization and the role of the cost function in neural network training, consider the task of weighing exactly 380 g of flour. One begins by pouring an arbitrary amount onto the scale. The first measurement is likely either above or below the desired weight. To correct this, one gradually adds or removes spoonfuls of flour until the precise weight is reached. This process may require multiple adjustments (analogous to epochs) before minimizing the error (difference from the target) and achieving the optimal result.

The effectiveness of a trained model is not determined only by how much the loss decreases on the training set, but mainly by its ability to generalize, i.e., to maintain good performance on previously unseen inputs. In this context, optimization refers to the capability of the training algorithm to reliably reduce the loss and converge toward a suitable set of parameters, while generalization describes the robustness of the learned mapping with respect to realistic input variations. Overfitting occurs when the model adapts excessively to the training set, including noise and incidental patterns, leading to degraded performance on validation or test data.

Generalization is primarily influenced by the amount and representativeness of the available data, the model capacity (architecture and number of parameters), and the intrinsic complexity of the task. While the problem complexity is not controllable, generalization can be improved either by increasing and diversifying the training data or by appropriately constraining model capacity through architectural choices and regularization.

In practice, overfitting is detected by monitoring training and validation performance. A typical symptom is a widening gap between training and validation loss: training continues to improve while validation deteriorates. Common countermeasures include using more data, simplifying the model (reducing the number of parameters), applying regularization (e.g., dropout or L2 weight decay), early stopping (interrupting training when validation loss starts increasing), and data augmentation (expanding the dataset through small, realistic transformations of the original samples).

Backpropagation (short for “backward propagation of error”) is the central technique for optimizing neural networks. It computes how changes in weights and biases affect prediction accuracy, by evaluating the gradient of the error with respect to the network parameters (Cilimkovic, 2015).

The parameter updates depend not only on the error but also on the learning rate which controls the speed of updates.

Parameter updates can be performed using different optimization algorithms. The simplest approach is Gradient Descent, where parameters are iteratively moved in the direction that most reduces the loss. In many practical applications, adaptive methods such as Adam are preferred, as they automatically adjust effective learning rates for individual parameters based

on past gradient information, often improving stability and speeding up convergence.

Despite these tools, training remains challenging. High-quality and sufficiently large datasets are essential to obtain robust models and reliable generalization. From an optimization perspective, convergence may be hindered by local minima or flat regions of the loss landscape. In addition, backpropagation can suffer from vanishing or exploding gradients, which can slow down learning or destabilize updates. Training time and computational resources may also become significant constraints. Finally, hyperparameter selection (e.g., learning rate, number of layers, and number of neurons) strongly influences performance and typically requires careful tuning.

Table I.1 *Summary of key training concepts and practical levers in ANNs.*

Concept	Meaning in training	Typical practical use
Optimization	Iterative minimization of the loss via parameter updates	Optimizer choice (GD/Adam) and learning-rate tuning
Generalization	Ability to perform well on unseen data	Validation-based model selection; control of model capacity
Overfitting	Training fits noise and loses generalization	Reduce parameters; regularization; early stopping
Regularization	Penalizes excessive model flexibility	Dropout; L2 weight decay
Early stopping	Stops training before overfitting increases	Monitor validation loss and stop when it rises
Data augmentation	Artificially increases data variability	Small, realistic transformations/perturbations
Gradient issues	Vanishing/exploding gradients hinder learning	Initialization/architecture/ activation choices
Computational constraints	Training time and resources limit experiments	Hardware choice; efficient training pipeline

I.2 Measurement Uncertainty in ANNs

The estimation of measurement uncertainty in the field of Artificial Intelligence is of fundamental importance, as it provides a means to assess and quantify the reliability of such models (Kanal & Lemmer, 2014). Without a proper evaluation of uncertainty, the predictions and decisions made by AI systems risk being opaque and potentially misleading (Psaros et al., 2023), especially in critical applications where accuracy and trustworthiness are essential.

According to the Guide to the Expression of Uncertainty in Measurement (GUM), uncertainty is defined as a "parameter, associated with the result of a measurement, that characterizes the dispersion of the values that could

reasonably be attributed to the measurand" (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, 2008). Greater dispersion indicates lower confidence in the result. The GUM framework distinguishes between Type A evaluations, which rely on repeated observations, and Type B evaluations, which employ mathematical modeling when repeated measurements are impractical.

In ANNs, two main categories of uncertainty are commonly considered: *aleatoric* uncertainty AU, arising from variability in the input data, and *epistemic* uncertainty EU, stemming from limitations in the model itself (Shirmohammadi, S., & Al Osman, H., 2021).

I.2.1 Aleatoric Uncertainty

Aleatoric uncertainty, also known as statistical uncertainty, refers to the intrinsic variability present in data. It arises from the stochastic nature of the underlying phenomena and cannot be reduced or eliminated (Nguyen et al., 2019), even with larger datasets or more sophisticated models. In other words, it is a property of the data-generating process itself rather than the model used to analyze it (Hüllermeier & Waegeman, 2021). This makes aleatoric uncertainty unavoidable.

A simple illustration is measurement noise: even the most precise sensors inevitably produce small fluctuations in their readings. A more concrete example is given by Bachstein (2019). Consider a regression problem where the goal is to predict the impact location of an arrow based on three variables: the force applied to the bowstring and two angles defining the shooting direction. In an idealized setting, with perfect knowledge of these variables, the model could predict the exact point of impact. In reality, however, two shots performed with identical force and angles will not land in precisely the same spot. The variability comes from uncontrollable external influences such as wind, changing atmospheric conditions, slight inconsistencies in the archer's technique, or small imperfections in the arrows. These factors, often ignored or simplified in the model, introduce fluctuations that cannot be eliminated by collecting more data or refining the model. For this reason, rather than predicting a single deterministic impact point, it is more appropriate to adopt a probabilistic perspective that captures the range of possible outcomes. Evaluating and understanding aleatoric uncertainty is essential for making realistic predictions and informed decisions. By accounting for irreducible variability, models become more robust and better suited to handle the inherent unpredictability of real-world data.

I.2.2 Epistemic Uncertainty

Epistemic uncertainty arises when a model cannot fully capture the relationship between inputs and outputs due to limited or imperfect

information (Hüllermeier & Waegeman, 2021). It can be reduced by gathering more or higher-quality data or by improving the model itself (Kendall & Gal, 2017). In the literature, this uncertainty is often described in terms of different aspects: the lack of knowledge about which model is most appropriate for the problem; the limitations that come from the fact that learning algorithms can only approximate the ideal model depending on data quality, diversity, and model capacity; and the issues caused by gaps or biases in the dataset, such as systems trained only in specific conditions or on restricted populations performing poorly when faced with new situations.

A practical example is found in autonomous driving. Consider a perception system trained to detect pedestrians. If the training dataset lacks images of pedestrians at night or in unusual weather conditions (e.g., heavy fog or snow), the system may exhibit high epistemic uncertainty in these situations. Unlike sensor noise (aleatoric uncertainty), this uncertainty could be reduced by expanding the dataset to include such scenarios or by adopting a more expressive model architecture. Quantifying epistemic uncertainty is particularly valuable in applications such as active learning, where the goal is to identify which data points should be labeled to most effectively improve the model. Conversely, failing to estimate epistemic uncertainty can have serious consequences in safety-critical domains like healthcare or autonomous systems, where overconfident but incorrect predictions can lead to dangerous decisions.

Figure I.9 provides a graphical illustration of the distinction between aleatoric and epistemic uncertainties.

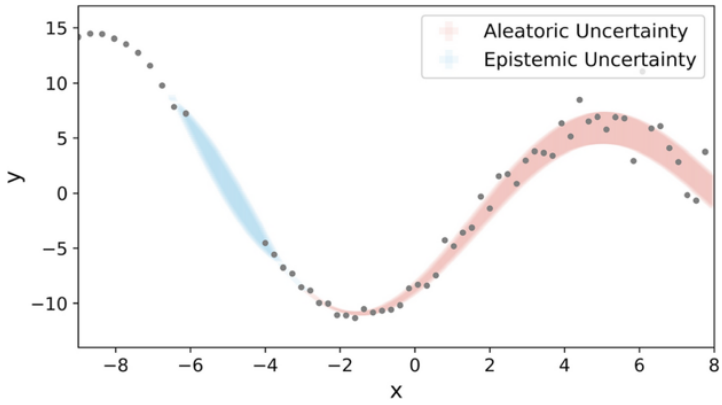

Figure I.9 *Difference between random and epistemic uncertainty (Yang & Li, 2023).*

I.2.3 Predictive Uncertainty

As reported in (Abdar et al., 2021), the combination of epistemic uncertainty and aleatoric uncertainty is called predictive uncertainty:

$$PU = EU + AU. \tag{I.9}$$

Predictive uncertainty is a central concept in machine learning (Tyralis & Papacharalampous, 2024), and particularly in Artificial Neural Networks (Lakshminarayanan et al., 2017). It captures the limited knowledge we have about a phenomenon or a model and naturally arises from the learning process: models are trained on incomplete and imperfect data, yet they are expected to generalize and make predictions on new, unseen inputs.

Understanding and quantifying predictive uncertainty is crucial for several reasons (Gawlikowski et al., 2023):

- Assessing reliability - Estimating uncertainty allows us to gauge when a model's predictions can be trusted. This is especially important in high-stakes domains, where incorrect outputs may endanger human life, cause financial loss, or lead to system failures.
- Supporting decision-making - In contexts where multiple choices involve different levels of risk, uncertainty acts as a guide, helping stakeholders make informed and safer decisions.
- Improving generalization - Models that explicitly account for uncertainty can better handle unfamiliar or challenging data. By identifying predictions with low confidence, they can adapt their behavior, reduce errors, and achieve greater robustness.
- Guiding data acquisition - Uncertainty can be used to drive active learning, where new labeled data are collected selectively from the most uncertain samples. This targeted approach improves performance efficiently by focusing resources where they matter most.

In summary, predictive uncertainty is not merely a theoretical construct but a practical mechanism that enhances the safety, reliability, and efficiency of machine learning systems. By integrating uncertainty into ANN-based models, these systems evolve from opaque predictors into interpretable and certifiable components of complex decision-making pipelines. Ultimately, uncertainty provides a more comprehensive description of prediction quality-analogous to a measurement in metrology-assuming that all systematic effects in the process have been identified and corrected.

I.2.4 Uncertainty Quantification Methods

The evaluation of predictive uncertainty has been explored in the literature through a variety of methodologies (Zhang et al. 2020). The next section outlines the principal approaches commonly used for uncertainty assessment. In addition to the most popular UQ families (Bayesian methods and ensembles), it is important to highlight a line of work that is particularly close to the goal of this thesis: measurement (input) uncertainty propagation through a trained neural network. In this setting, the network is treated as a deterministic mapping and the objective is to propagate a known uncertainty on the inputs to obtain the induced uncertainty on the outputs, in a way that is conceptually aligned with ISO-GUM.

I.2.4.1 Bayesian Methods

Bayesian methods are among the most widely adopted approaches in the literature for uncertainty estimation. A key advantage of these methods is their ability to capture both epistemic uncertainty and aleatoric uncertainty. The central idea is to assign probability distributions not only to the network weights but also to the inputs and outputs, thereby framing the entire learning and prediction process within a probabilistic setting (Mara et al., 2016) (Figure I.10). In principle, this approach naturally leverages the law of total probability (Gal & Ghahramani, 2015), allowing a rigorous treatment of uncertainty. However, when applied to DNNs, Bayesian methods face major challenges. The sheer number of parameters and the scale of modern datasets make exact Bayesian inference computationally infeasible and analytically intractable. To address this limitation, researchers often resort to approximate inference schemes that provide Bayesian-motivated uncertainty estimates without requiring fully probabilistic Bayesian neural networks. One of the most popular techniques in this context is Monte Carlo Dropout (MCD) (Gal & Ghahramani, 2016). While MCD is frequently discussed within the Bayesian deep learning literature, it should be viewed as an approximation to variational Bayesian inference rather than a fully Bayesian treatment of neural networks.

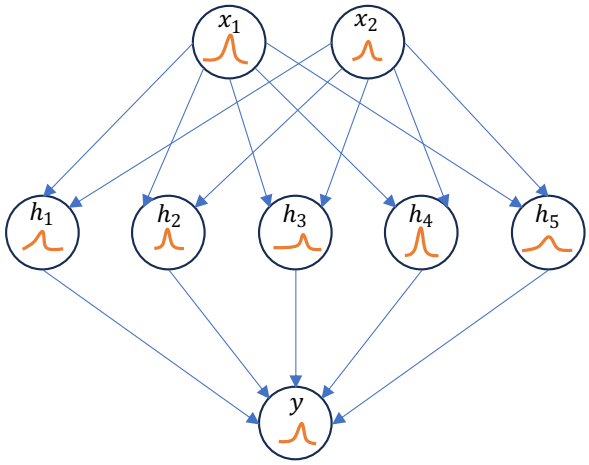

Figure I.10 *Conceptual illustration of a Bayesian neural network: inputs and layer activations are treated probabilistically, and uncertainty is propagated through the network to obtain a predictive distribution for the output y.*

MCD operates by exploiting the dropout mechanism, which was originally proposed as a regularization technique (Srivastava et al., 2014) to reduce overfitting during training. Unlike its conventional use, where dropout is disabled at test time, MCD keeps dropout active during inference and performs multiple stochastic forward passes through the network (Figure I.11). By aggregating the resulting predictions $\{\hat{y}^{(t)}\}_{t=1}^{T}$, MCD builds an empirical approximation of the predictive distribution: the sample mean $m(x)$ provides a point prediction and the sample variance $\sigma^2(x)$ quantifies predictive uncertainty (Milanés-Hermosilla et al., 2021).

It works by randomly deactivating a subset of neurons at each training step, forcing the network to rely on different combinations of neurons and preventing over-dependence on specific connections. In Monte Carlo Dropout, dropout is deliberately kept active during inference. Multiple stochastic forward passes are performed for the same input, each time with different neurons randomly dropped out. This produces a distribution of outputs rather than a single deterministic prediction. The variability among these outputs captures the model's uncertainty, allowing for principled and computationally efficient uncertainty quantification. From this empirical distribution, the predictive mean $m(x)$ can be used as the final output, while the variance (or standard deviation σ) summarizes the uncertainty, consistent with the spread illustrated in Figure I.11 (e.g., $\pm 2\sigma$ under a Gaussian approximation). Regions of the input space well-covered by training data generally yield low variance, while regions with sparse or unseen data produce higher variance, reflecting the model's lack of knowledge.

Monte Carlo Dropout is not a fully Bayesian neural network in the strict sense, because it does not explicitly place a prior over the weights and learn a posterior distribution through Bayesian inference. Instead, it provides a practical approximation to Bayesian model averaging that is closely related to variational Bayesian approaches. In particular, keeping dropout active at test time and repeating stochastic forward passes can be interpreted as sampling from a variational approximation in which dropout induces a Bernoulli variational distribution over the network weights (Gal & Ghahramani, 2015; Li & Gal, 2017). Under this interpretation, the empirical mean and variance of the sampled predictions approximate the posterior predictive mean and uncertainty, offering an efficient proxy for epistemic uncertainty.

This distinction is important: MCD offers a convenient approximation, whereas fully Bayesian neural networks explicitly represent weight uncertainty and require more demanding inference procedures.

Despite this, MCD has been widely applied across domains, including regression, classification, image segmentation, and anomaly detection. In summary, Monte Carlo Dropout provides a versatile and practical approach to uncertainty quantification, combining regularization techniques with Bayesian principles to improve the reliability of neural network predictions.

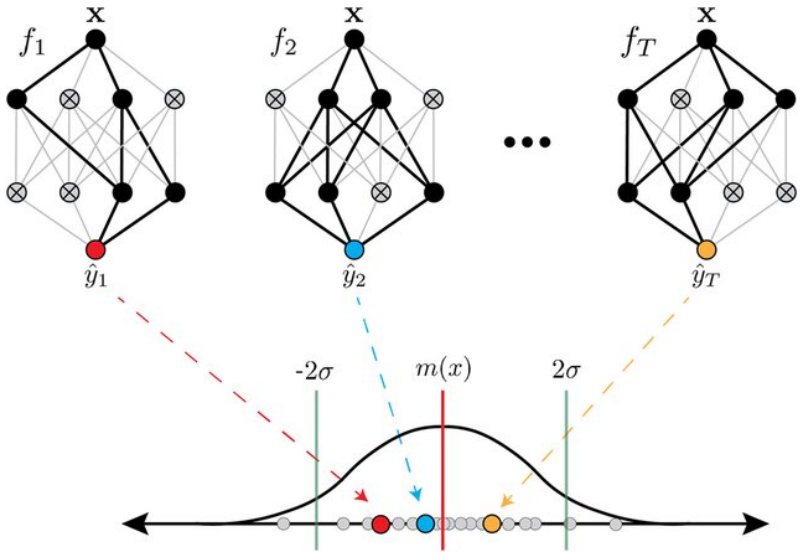

Figure I.11 *Monte Carlo Dropout (MCD) at inference time: dropout is kept active and the same input is propagated through the network T times, yielding stochastic predictions. Their empirical distribution provides an approximation of the predictive distribution; the sample mean is used as point prediction and the spread provides an uncertainty estimate (Van Katwyk et al., 2023).*

I.2.4.2 Ensemble Methods

Ensemble methods are widely used to enhance the performance and reliability of neural networks (Abdullah et al., 2024). Originally, both Bayesian and ensemble approaches were designed to improve predictive accuracy. Despite differences in implementation, they share the common principle of providing a probability density function (PDF) over predictions. For example, MCD generates multiple stochastic predictions from a single model by applying dropout during inference, whereas ensembles rely on multiple independently trained models whose outputs are aggregated to improve accuracy and quantify uncertainty (Hansen & Salamon, 2002). The defining feature of ensemble methods is their ability to combine predictions from multiple networks into a single mapping function. This aggregation can be performed in various ways, with common strategies including bagging, boosting, and stacking. Although these methods were not explicitly designed for uncertainty quantification, their inherent ability to generate diverse models makes them valuable for this purpose. The variability across predictions reflects predictive uncertainty:

- Bagging generates diversity by training models on bootstrapped subsets of the dataset, with uncertainty inferred from the spread of predictions.
- Boosting builds models sequentially, where each subsequent learner focuses on difficult-to-predict samples, implicitly highlighting uncertain regions of the feature space.
- Stacking employs a meta-learner to combine the outputs of multiple base models, leveraging complementary strengths to provide more robust predictions and richer uncertainty estimates.

Figures I.12-I.14 illustrate the training/combination strategies; for uncertainty quantification, the key step is to retain the individual predictors' outputs and interpret their dispersion as an empirical predictive distribution rather than using only the aggregated estimate.

I.2.4.3 Bagging

One of the earliest ensemble techniques, bagging (Bühlmann, 2011), creates multiple training datasets by sampling with replacement from the original dataset (Figure I.12). Independent models are trained on these subsets, and their predictions are aggregated, via majority voting in classification tasks or averaging in regression. This aggregation step is central to bagging's ability to improve generalization and stability.

From an uncertainty-quantification perspective, bagging naturally provides a set of predictions for the same input x, one from each bootstrap-

trained model. The final estimate is typically obtained by averaging (regression) or majority probability averaging (classification), while the spread across the individual predictions can be used as a proxy for epistemic uncertainty (e.g., sample variance in regression or predictive entropy of the averaged class probabilities in classification).

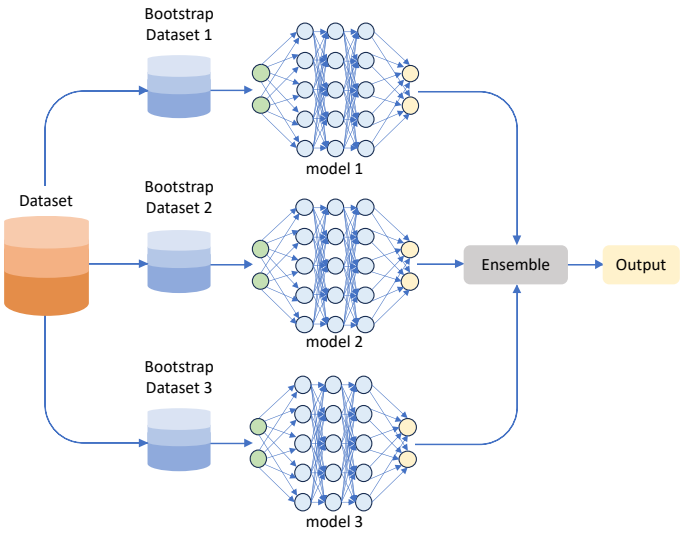

Figure I.12 *Bagging Ensemble mechanism (Brownlee, 2021).*

I.2.4.4 Boosting

In contrast to bagging's parallel approach, boosting (Bühlmann, 2011) builds models sequentially (Figure I.13). The first model is trained on the entire dataset, and subsequent models focus increasingly on the samples misclassified by earlier ones. By iteratively refining the decision boundary, boosting enhances predictive accuracy, particularly in challenging regions of the feature space.

Although boosting is primarily designed to improve accuracy, it still yields a collection of base learners whose combined output defines the final predictor. For uncertainty quantification, a practical (heuristic) indicator can be obtained by examining the variability of predictions across boosting stages or the magnitude of successive corrections: inputs that require large or unstable refinements across the sequence typically correspond to harder regions of the feature space and can be flagged as more uncertain.

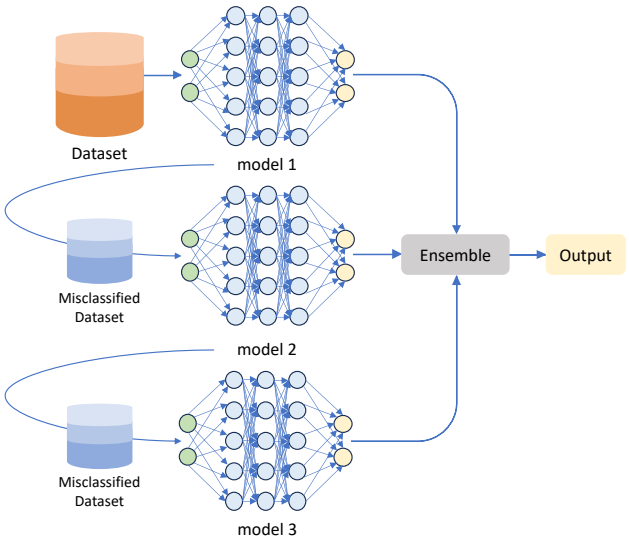

Figure I.13 *Boosting Ensemble mechanism (Brownlee, 2021).*

I.2.4.5 Stacking

Stacking also draws on the bootstrap principle but introduces a meta-model that learns how to optimally combine the outputs of base learners (Figure I.14). This hierarchical structure enables stacking to capture higher-order relationships among models, often leading to superior predictive performance (Odegua, 2019).

In stacking, uncertainty information can be extracted by keeping the base learners' individual outputs, since their disagreement reflects model uncertainty. The meta-learner combines these outputs into a final prediction, and when it produces calibrated probabilities, uncertainty can be summarized directly from the resulting predictive distribution (e.g., via entropy in classification).

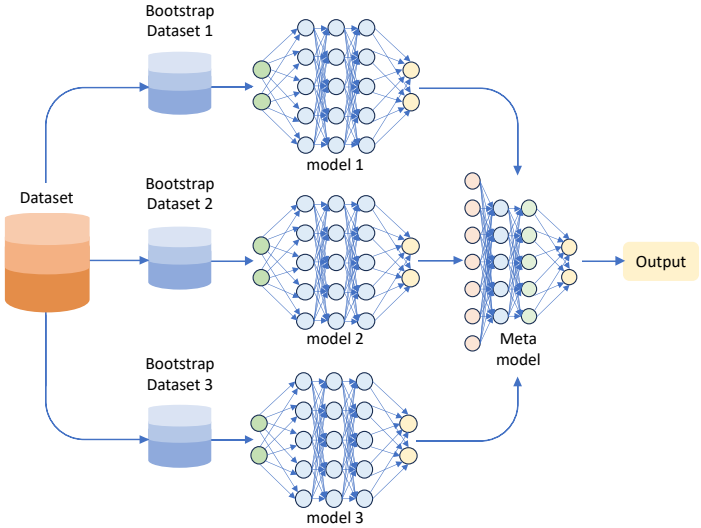

Figure I.14 *Stacking Ensemble mechanism (Brownlee, 2021).*

I.2.4.6 Single Deterministic Methods

Single Deterministic Methods encompass all approaches where predictive uncertainty is estimated using only a single forward pass through a deterministic neural network. Unlike Bayesian or ensemble methods, which rely on multiple models or stochastic sampling, these methods directly infer uncertainty from the internal representations of one trained model.

A prominent example is provided by Radial Basis Function (RBF) networks, which estimate uncertainty during a single inference step (Ghosh & Nag, 2001). In these models, prediction is guided by the concept of class centroids: representative feature vectors learned during training that correspond to different classes. Given an input sample, the model produces a feature representation f_θ. Prediction is then obtained by evaluating both a kernel function and a distance function between f_θ and the centroids c_k. Uncertainty is expressed as the distance between the predicted feature vector and the nearest centroid (Figure I.15). Intuitively, if the representation of a new input lies close to one of the class centroids, the model is confident in its classification. Conversely, if the feature vector is far from all centroids, the input is considered to fall out of distribution (OOD) (Liu et al., 2021), and the prediction is flagged as uncertain.

Beyond RBF-inspired approaches, deterministic uncertainty estimation methods are often categorized into internal and external approaches (Kabir et al., 2018; Abdar et al., 2021; Gawlikowski et al., 2023):

- Internal Uncertainty Quantification: The neural network is explicitly trained to produce both predictions and uncertainty estimates. Uncertainty modeling is integrated into the training procedure, influencing the learned representations and final outputs. Typical examples include learning distributional parameters (e.g., predictive variance) jointly with the output during training, as in heteroscedastic regression models (Kendall & Gal, 2017)
- External Uncertainty Quantification: Uncertainty is estimated without altering the original trained model. Typically, a second model is introduced to process the outputs of the predictive network and estimate the reliability of its predictions. This approach leaves the prediction mechanism untouched but requires additional parameters or coefficients to capture uncertainty. Representative examples include post-hoc confidence/uncertainty estimation based on feature-space density or distance to training representations, often used to flag out-of-distribution inputs (Van Amersfoort et al., 2020; Liu et al., 2021).

In both cases, the methodology often involves two parallel neural networks: one network for generating predictions, and another for quantifying the uncertainty associated with those predictions (Kabir et al., 2018; Abdar et al., 2021).

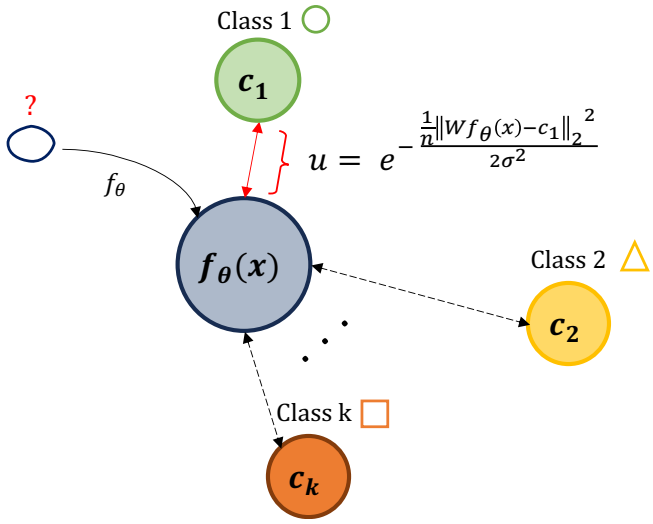

Figure I.15 *Deterministic uncertainty quantification (Adapted from Van Amersfoort et al., 2020).*

I.2.4.7 Analytical input uncertainty propagation (measurement-uncertainty perspective)

Besides confidence- and OOD-oriented deterministic methods, another stream of research focuses on propagating input (measurement) uncertainty through a trained network, viewed as a deterministic mapping $y = f_\theta(x)$. The aim is to estimate the output distribution (or, more commonly, its moments) induced by a known distribution of the inputs. This perspective is conceptually close to the ISO-GUM notion of uncertainty propagation in indirect measurements, where uncertainty on the measurand is obtained by propagating the uncertainty of the inputs through a measurement model.

Early work in robust speech recognition investigated uncertainty propagation through MLPs and DNNs when input features are affected by noise or distortion, relying on approximate propagation and sampling-based approximations (Astudillo & Silva Neto, 2011; Abdelaziz et al., 2015). In other application domains, efficient schemes have been proposed to propagate activation uncertainties through deep models using lightweight probabilistic approximations (Gast & Roth, 2018). More recent contributions derive analytical or quasi-analytical propagation rules under Gaussian assumptions and/or layer-wise linearization, resulting in closed-form (or near closed-form) expressions for output moments (Zhang et al., 2025; Diamzon & Venturi, 2026). Extended Kalman filtering has also been explored as an efficient moment-propagation mechanism for deep networks (Titensky et al., 2018).

To relax the Gaussian assumption and to approximate the full output probability density function, Monchot et al. (2023) propose propagating a Gaussian Mixture Model (GMM) through the network via a Split&Merge strategy. From a metrology-oriented viewpoint, Ludwig et al. (2025) explicitly connect neural-network uncertainty propagation to ISO-GUM principles and introduce the QuadLU activation function to simplify the computation of derivatives needed by analytical propagation. Finally, uncertainty propagation has been formulated using factor graphs, enabling inference/optimization-based propagation that can better accommodate complex architectures (e.g., skip connections) compared with purely feed-forward layer-by-layer propagation (Daruna et al., 2023).

These approaches are closer to the contribution of this thesis than Bayesian or ensemble techniques because they explicitly target measurement-driven (input) uncertainty propagation. The method proposed in Chapter II builds on this family by providing an ISO-GUM-compliant analytical framework tailored to the networks and tasks considered in this work, and by explicitly linking propagated uncertainty to the final decision stage.

Table I.2 *Representative approaches for input/measurement uncertainty propagation in neural networks.*

Family	References	Main assumptions	What is propagated	Output
Gaussian + linearization / quasi-linearization	Astudillo and Silva Neto (2011); Abdelaziz et al. (2015); Zhang et al. (2025); Diamzon and Venturi (2026)	Often Gaussian inputs; local linearization of nonlinearities	Moments (mean/variance), sometimes approximate PDFs	Output moments (and sometimes approximate PDFs)
Assumed-density filtering/ lightweight probabilistic propagation	Gast and Roth (2018)	Approximate distributions per layer	Activation uncertainties	Predictive mean + uncertainty
EKF-based propagation	Titensky et al. (2018	Mild assumptions on input distribution; EKF approximations	Mean/covariance	Output moments (+ optional model error term)
Non-Gaussian propagation (GMM)	Monchot et al. (2023)	GMM representation; split/merge control	Full input distribution approximation	Output PDF estimate
Metrology-oriented (GUM, derivative-friendly design)	Ludwig et al. (2025)	Focus on derivative availability and propagation	Moments via analytical propagation	Output uncertainty consistent with GUM framing
Graphical models/factor graphs	Daruna et al. (2023)	Inference/optimization view; supports complex graphs	Uncertainty through network graph	Output uncertainty (aleatoric-focused)

I.2.4.8 Test Time Data Augmentation Methods

Test-Time Data Augmentation (TTDA) is one of the most straightforward approaches for estimating aleatoric uncertainty in deep learning models (Ayhan & Berens, 2018). The method relies on the intuition that applying augmentations to test samples exposes different aspects of the data, thereby allowing the model's sensitivity to variations to be explored. The procedure is simple: for each test input, multiple augmented variants are generated using

standard data augmentation techniques (e.g., random crops, flips, rotations, color perturbations) (Wang et al., 2018). Each variant is passed through the model, producing a set of predictions. By aggregating these predictions, a predictive distribution is obtained, from which both the final output and an uncertainty estimate can be derived (Figure I.16).

This technique offers several practical advantages:

- Model-preserving: It does not require retraining or modifying the original model.
- Data-efficient: No additional training data is needed.
- Versatile: Applicable to a wide range of tasks, from image classification to natural language processing, as long as suitable augmentations can be defined.

However, the reliability of TTDA depends critically on the choice of augmentations. To ensure meaningful results, only in-distribution augmentations (transformations that preserve the semantic content of the input) should be applied. For instance, flipping an image of a dog may still yield a valid example, whereas rotating handwritten digits by 180° could produce samples outside the intended distribution.

A notable challenge is that TTDA can sometimes destabilize predictions:

- Correct predictions may become incorrect under certain augmentations.
- Incorrect predictions may turn correct if augmentations happen to align better with learned features.

The degree of variability depends on factors such as the task complexity, the size and quality of the training dataset, the architecture of the neural network, and the specific augmentations chosen. Because of this, one of the key open questions in TTDA is identifying which augmentations are most informative for a given task and how many augmented samples are necessary to obtain reliable uncertainty estimates. Overly aggressive or inappropriate augmentations may artificially inflate uncertainty, while too few augmentations may fail to capture it adequately.

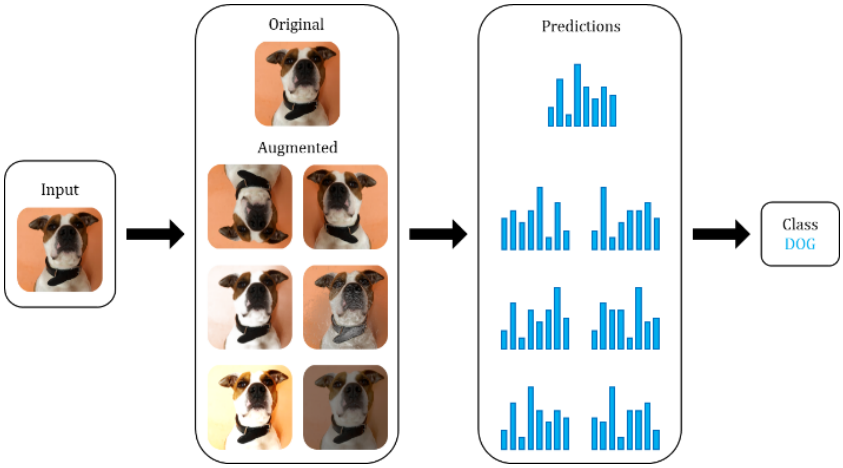

Figure I.16 *Test Time Data Augmentation (Adapted from https://stepup.ai/test_time_data_augmentation/).*

I.2.4.9 Brief comparison of methods

All the methods described approach the problem of uncertainty quantification in different ways, each offering its own advantages and limitations. While some techniques may prioritize analytical rigor, others emphasize computational efficiency or flexibility in handling complex models. The choice of method therefore often depends on the specific application, the type of data available, and the balance between accuracy and computational cost. Below, the main features and differences of these methods are summarized and compared.

- Bayesian methods quantify primarily epistemic uncertainty. By introducing stochasticity in the weights and activations, they produce a PDF of outputs that reflects variability in the learned model. However, from a metrological standpoint, this procedure resembles the behavior of model ensembles rather than the propagation of measurement uncertainty, since each iteration modifies the neural interconnections and effectively alters the system architecture. Computationally, Bayesian approaches are highly demanding: both training and inference require multiple stochastic forward passes, leading to significantly longer training times and slower evaluations compared to other techniques.
- Ensemble methods also mainly capture epistemic uncertainty. By training multiple independent networks and combining their outputs, ensembles improve predictive performance and generate more stable estimates of uncertainty. In practice, the diversity among the models allows the ensemble to reflect the confidence

associated with the learned function. However, the conditions under which ensembles yield reliable uncertainty estimates are not fully understood, and their theoretical link to metrological uncertainty is weak. From a computational perspective, ensembles are very costly: training requires training several networks from scratch, and evaluation demands a number of forward passes proportional to the ensemble size. This makes them less practical for large-scale or real-time applications.

- Single deterministic methods are a heterogeneous category. Many approaches estimate confidence via internal representations and are primarily used for epistemic/OOD-related assessments; however, a dedicated subset of deterministic techniques explicitly targets measurement-driven aleatoric uncertainty by propagating a known input distribution through the trained network (see Section I.2.4.7).
- Among data-augmentation-based techniques, TTDA is conceptually aligned with measurement uncertainty propagation in an empirical sense, as it generates an output distribution by repeatedly perturbing the input. Nevertheless, there also exist dedicated analytical propagation methods (Section I.2.4.7) that address input/measurement uncertainty more directly and can be viewed as even closer to the ISO-GUM perspective. However, analytical input uncertainty propagation methods are even closer to the ISO-GUM perspective because they explicitly propagate a prescribed input distribution (see Table I.2).

Table I.3 summarizes the main UQ families in terms of the uncertainty type they address and their computational demands. It is worth noting that most popular approaches focus on predictive uncertainty by injecting variability in the model or by sampling at inference time, and they often address input uncertainty only indirectly. In contrast, measurement-oriented methods explicitly propagate a prescribed input uncertainty through a trained network, which is conceptually closer to the ISO-GUM measurement-model view. Representative analytical propagation approaches and their assumptions are summarized in Table I.2.

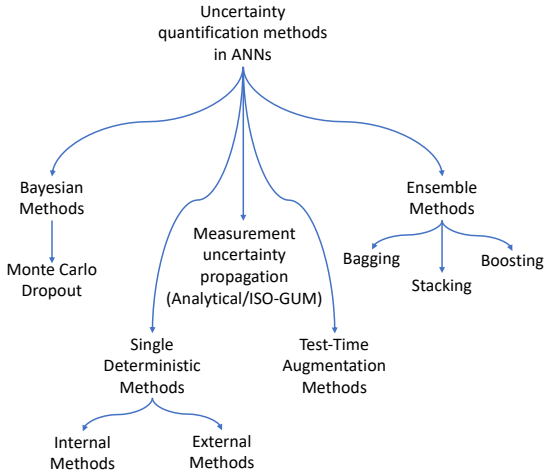

Figure I.17 *Overview of uncertainty quantification approaches in ANNs. In addition to popular predictive-uncertainty families, the diagram highlights measurement-oriented input uncertainty propagation methods (analytical/ISO-GUM-related), whose representative techniques and assumptions are summarized in Table I.2.*

Table I.3 *Comparison on the state-of-the-art methodologies for uncertainty estimation in ANNs.*

	Bayesian Methods	**Ensemble Methods**	**Single Deterministic Methods**	**Test-Time Augmentation Methods**	**Analytical**
Evaluation of epistemic uncertainty	*Yes*	*Yes*	*Yes*	*No*	*No*
Evaluation of aleatoric uncertainty	*Yes*	*Limited*	*It depends on the method*	*Yes*	*Yes*
Use of different networks	*No (MC Dropout)*	*Yes*	*It depends on the method*	*No*	*No*
Number of inputs per prediction	*1*	*1*	*1*	*Several*	*1*
Computational effort during training	*High*	*High*	*Low*	*Low*	*Low*
Computational effort during inference	*High*	*High*	*Low*	*High*	*Low-Medium*
Alignment with ISO-GUM	*Weak*	*Weak*	*Weak*	*Partial*	*Strong*

This entry mainly refers to deterministic confidence methods; deterministic input uncertainty propagation approaches are summarized separately in Section I.2.4.7.

The current state of the art primarily focuses on quantifying model uncertainty by introducing variability within the model and, to some extent, analyzing the contribution of aleatoric uncertainty. However, these approaches often rely on indirect techniques, as the uncertainty associated with the input side is not explicitly addressed. This limitation highlights the need for a methodology grounded in the ISO-GUM.

In the following chapter, we propose an approach that integrates GUM principles into Artificial Neural Networks, establishing a standardized and metrologically rigorous framework for aleatoric uncertainty quantification. This approach not only provides a more comprehensive assessment of uncertainty, including input-related contributions, but also enhances the interpretability and reliability of network predictions.

Chapter II
Aleatoric Uncertainty Quantification: An ISO-GUM Approach

Throughout this chapter, the measurement uncertainty concepts and propagation formulas follow the ISO-GUM framework (JCGM 100:2008) and its Supplement 1 for Monte Carlo propagation of distributions (JCGM 101:2008).

II.1 Law of Propagation of Uncertainty LPU

According to the GUM, when a measurement result cannot be obtained directly but is instead determined from the measurements of several other quantities, the evaluation of its uncertainty must be carried out using the law of propagation of uncertainty (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, 2008). This approach provides a systematic framework for quantifying the uncertainty associated with the output quantity by analyzing how the uncertainties of the input quantities propagate through the mathematical relationship that links them.

In particular, GUM establishes that if the measurand y is defined by a known function f of N input variables

$$y = f(x_1, x_2, \dots, x_N), \tag{II.1}$$

then the uncertainty of y can be evaluated by considering both the uncertainties associated with each input quantity x_{i} and the sensitivity of the function f with respect to variations in these inputs (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, 2008). This formulation ensures that the

combined standard uncertainty of the measurand reflects not only the magnitudes of the input uncertainties but also the way in which the functional dependence amplifies or attenuates their effects.

The measurement uncertainty can therefore be evaluated using the following expression:

$$u_y{}^2 = \sum_{i=1}^{N} \left(\frac{\partial f}{\partial x_i}\right)^2 u_{x_i}{}^2. \tag{II.2}$$

Each u_{x_i} is a standard uncertainty evaluated as a Type A or Type B uncertainty (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, 2008).

In the ISO-GUM framework, uncertainty propagation requires that the input quantities are associated with standard uncertainties and (when needed) with an assumed probability distribution. In practice, the input standard uncertainty $u(x_i)$ can be obtained either from repeated measurements (Type A) or from non-statistical information (Type B), such as instrument specifications, calibration certificates, or datasheets.

For Type A evaluation, if n repeated measurements of an input quantity are available, the standard uncertainty of the estimated mean can be obtained from the sample standard deviation $s(x_i)$ as $u(x_i) = s(x_i)/\sqrt{n}$.

For Type B evaluation, the uncertainty is derived from available bounds or specifications by assuming a distribution. Common conversions include:

- A tolerance $\pm a$ modeled as a rectangular distribution, which yields $u(x_i) = a/\sqrt{3}$;
- A quantization step or resolution r modeled as rectangular over $\pm r/2$, which yields $u(x_i) = r/\sqrt{12}$;
- A specification given as an expanded uncertainty U with coverage factor k, which yields $u(x_i) = U/k$.

Once $u(x_i)$ is defined, an input distribution must be specified for Monte Carlo propagation. In this thesis, unless otherwise stated in the experimental chapters, each input quantity is modeled as Gaussian, $X_i \sim \mathcal{N}(x_{i,0}, u^2(x_i))$, where $x_{i,0}$ is the nominal value used by the deterministic forward pass. If correlations between inputs are relevant, they are represented through a covariance matrix and handled using the correlated form of the LPU.

Equation II.2 holds under the assumption that the input quantities x_i are uncorrelated, meaning that no statistical dependence exists among them. In this case, the combined uncertainty of the measurand can be evaluated simply by summing the contributions of each input uncertainty, weighted by the corresponding sensitivity coefficients $\frac{\partial f}{\partial x_i}$.

However, when the input quantities are correlated, this assumption no longer applies, and the law of propagation of uncertainty must be expressed

in a more general form that accounts for the covariances between the inputs. The counterpart of the previous equation for correlated input quantities is therefore:

$$u_y{}^2 = \sum_{i=1}^{N} \left(\frac{\partial f}{\partial x_i}\right)^2 u_{x_i}{}^2 + 2\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \frac{\partial f}{\partial x_i}\frac{\partial f}{\partial x_j} u(x_i, x_j), \tag{II.3}$$

where $u(x_i, x_j)$ denotes the covariance between the input quantities x_i and x_j (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, 2008). This formulation ensures that the combined standard uncertainty of the measurand properly reflects both the individual uncertainties of the input quantities and their mutual statistical correlations.

GUM also states that when f exhibits significant non-linearities, the above simplified formula cannot be applied directly (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, 2008). Instead, its more complete version including higher-order Taylor series terms must be used. Consequently, when dealing with neural networks, which are inherently non-linear models, it becomes necessary to use the following formula recommended by GUM for a non-linear f (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, 2008; BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, 2008b):

$$\begin{aligned} u_y{}^2 = \sum_{i=1}^{N} \left(\frac{\partial f}{\partial x_i}\right)^2 u_{x_i}{}^2 + 2\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} \frac{\partial f}{\partial x_i}\frac{\partial f}{\partial x_j} u(x_i, x_j) \\ + \sum_{i=1}^{N}\sum_{j=1}^{N}\left[\frac{1}{2}\left(\frac{\partial^2 f}{\partial x_i \partial x_j}\right)^2 \right. \\ \left. + \frac{\partial f}{\partial x_i}\frac{\partial^3 f}{\partial x_i \partial x_j{}^2}\right] u^2(x_i) u^2(x_j). \end{aligned} \tag{II.4}$$

But, applying the above formula to a neural network, which can be quite large, makes the analysis very complicated. Instead, an appropriate linearization approximation can be used, as done in other recent works (Diamzon & Venturi, 2025; Malmström, 2023; Zhang, Singh, Menolascino, & Ching, 2025).

In this thesis, the law of propagation of uncertainty is applied to trained neural networks by treating the network as a deterministic measurement model. Since the direct use of higher-order terms is impractical for large-scale networks, a first-order linearization strategy is adopted to enable analytical propagation. The resulting neural-network-specific propagation framework is developed in Section II.3.

When properly confined to this role, it provides an efficient means of quantifying how uncertainties propagate through nonlinear transformations, while ensuring that the predictive capability of the neural network remains intact.

Therefore, an algorithm is needed that emulates the behavior of the neural network and performs uncertainty propagation through the different nodes of the network based on the uncertainties associated with the input data, as illustrated in Figure II.1.

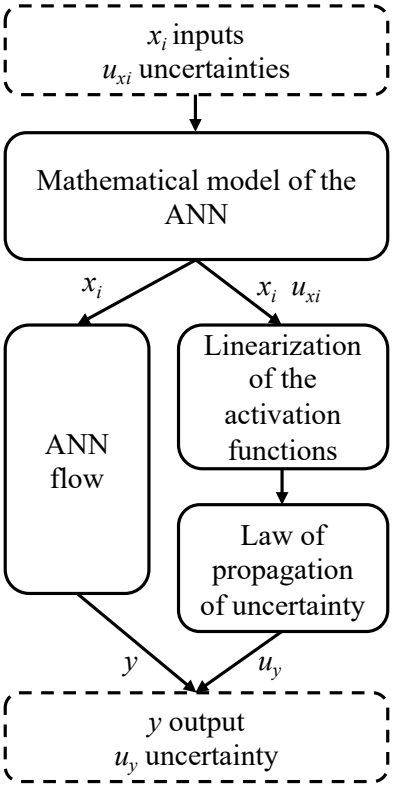

Figure II.1 *Flowchart of the proposed methodology.*

In regression problems, the propagated standard uncertainty can be interpreted directly as an uncertainty attached to the predicted output value. In classification, instead, the network output is typically a score or a class probability, and the propagated uncertainty can be exploited in two conceptually different ways.

- Probability/score calibration (decision-level use): the pair (μ, σ) is mapped to a single calibrated measure of class confidence (e.g.,

the probability that the uncertain score lies on the selected side of the decision threshold).

- Second-order uncertainty: (μ, σ) is interpreted as defining a distribution over the class score/probability itself (Gaussian in this work), from which confidence intervals and risk-aware decision rules can be derived.

The classification-oriented interpretation adopted in this thesis is detailed in Section II.3.

To assess the effectiveness and validity of the proposed methodology, it is essential to compare it against a well-established reference method. To validate the analytical uncertainty propagation developed in this chapter, Monte Carlo propagation of distributions is adopted as a numerical reference method, as recommended in GUM Supplement 1. The Monte Carlo procedure and its practical considerations are described in Section II.2, while Section II.4 clarifies how Monte Carlo estimates relate to the deterministic prediction and to the proposed LPU-based uncertainty propagation.

II.2 Monte Carlo Simulation

The Monte Carlo simulation quantifies uncertainty by explicitly accounting for the variability in input parameters. This is accomplished by repeatedly performing the prediction process, each time sampling input values from their respective probability distributions. This approach is particularly valuable for complex models where analytical propagation of uncertainty is difficult or infeasible (BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML, 2008b).

Its flexibility lies in the fact that it does not require detailed knowledge of the model's internal structure. For example, in neural networks, where input-output relationships can be highly nonlinear, Monte Carlo methods can effectively estimate uncertainty without explicitly computing sensitivity coefficients. However, these advantages come with trade-offs. The computational cost can be substantial, especially for deep or complex networks, since thousands of forward passes may be needed to obtain reliable uncertainty estimates.

In this work, Monte Carlo uncertainty propagation is implemented as follows. For a given nominal input x_0 and prescribed input distributions, Mindependent input realizations $\{x^{(j)}\}_{j=1}^{M}$ are drawn. Each sample is propagated through the trained network to obtain $\hat{y}^{(j)} = f_\theta(x^{(j)})$. The Monte Carlo estimate of the output mean is computed as $\hat{\mu}_y = \frac{1}{M}\sum_{j=1}^{M}\hat{y}^{(j)}$, while the output uncertainty is summarized by the sample standard deviation $\hat{\sigma}_y$ (or variance $\hat{\sigma}_y^2$).

Neural networks' nonlinearity further complicates this: they can exhibit saturation effects or flat regions where large input changes produce little

output variation, or conversely, small input perturbations can cause significant output fluctuations. Determining the number of Monte Carlo samples required for statistical convergence is also non-trivial and can demand extensive computation. Inadequate sampling may introduce bias, leading to under- or overestimation of uncertainty. While Monte Carlo simulations offer valuable insights into network output variability, these limitations must be carefully managed to ensure meaningful results. Nonlinear activation functions, such as ReLU, can introduce additional biases in output estimation. Input samples that fall within inactive regions of the function, for instance, negative values for ReLU, produce no contribution to the output, as ReLU maps these inputs to zero. This can distort the estimated mean and variance, skew the output distribution, and cause discrepancies between the Monte Carlo prediction and analytically expected output. Consequently, when interpreting Monte Carlo results for networks with activation functions featuring sharp transitions or dead zones, caution is required, as these effects may lead to misestimation of both the output and its uncertainty.

Having described Monte Carlo propagation and its practical limitations, the next section introduces the LPU-based probabilistic framework adopted in this thesis to interpret the propagated output uncertainty in classification tasks.

II.3 LPU-Based Probabilistic Framework for Classification

This section derives an analytical uncertainty propagation scheme for neural networks based on the first-order LPU introduced in Section II.1.

In general, neural networks are commonly regarded as black-box models, since their internal mechanisms are not readily interpretable and the mapping from inputs to outputs is often treated as opaque. The first challenge, therefore, lies in transforming the network into a white-box representation, that is, making its underlying mathematical structure explicit. This task is far from trivial, as it requires a detailed examination of the network architecture and is further complicated by the high dimensionality of the parameter space, which includes numerous weights and biases distributed across multiple layers.

The central idea of this work is to apply the principles established in the GUM to the mathematical model of a pre-trained neural network. To this end, it is necessary to first derive an explicit formulation of the functional relations that define the network's computations. Once this formulation is available, the nonlinear activation functions are appropriately linearized around suitable operating points, thereby enabling the application of the law of propagation of uncertainty. Through this approach, the network is no longer treated as a black box but instead as a transparent system in which the propagation of uncertainties from inputs and parameters to outputs can be rigorously quantified.

Figure II.2 provides a schematic depiction of the fundamental structure of a single neuron in a neural network model. The neuron receives multiple

inputs, each scaled by an associated weight that reflects the strength of the corresponding connection. These weighted inputs are then aggregated at a summation node, together with an additional bias term:

$$v_k = \sum_{i=1}^{N} (x_i w_{k,i}) + b_k. \tag{II.5}$$

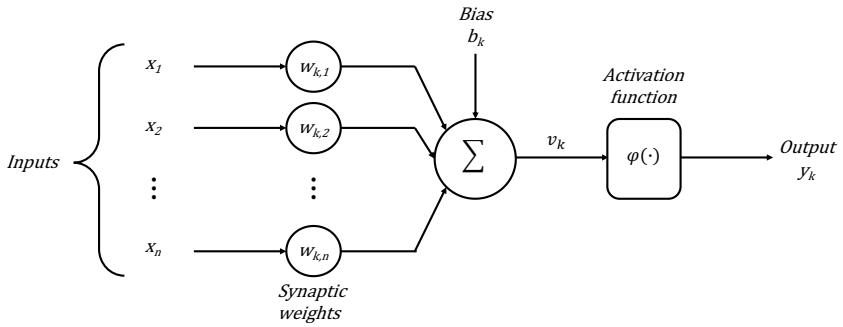

Figure II.2 *Artificial neuron structure.*

The resulting sum is then passed through an activation function, which constrains the output amplitude of the neuron:

$$y_k = \varphi(v_k). \tag{II.6}$$

By applying the LPU to the structure of an artificial neuron, it is possible to derive the uncertainty associated with its output. This output uncertainty is determined as a function of several contributing factors, including the uncertainties of the input variables, the synaptic weights, and the bias term. In this way, the method quantifies the individual contribution of each parameter to the total uncertainty, thereby providing a detailed characterization of how uncertainty propagates within the neural network. Specifically, the uncertainty of the neuron's intermediate value v_k can be evaluated as follows:

$$u_{vk}^2 = \sum_{i=1}^{N} \left(\frac{\partial v_k}{\partial x_i}\right)^2 u^2(x_i) = \sum_{i=1}^{N} w_{k,i}^2 u^2(x_i). \tag{II.7}$$

The subsequent step consists of propagating the estimated uncertainty through the activation function, which is generally nonlinear in the context of neural networks. To make the analysis tractable, a linearization of the activation function around an appropriate operating point is employed, as previously discussed. Specifically, the activation function is approximated by its first-order Taylor expansion around the operating point $\bar{v}_k$, where $\bar{v}_k$

denotes the nominal (deterministic) value of v_k obtained from the forward pass with nominal inputs and parameters.

Under this approximation, the expression for the output uncertainty becomes:

$$\varphi(v_k) \approx \varphi(\bar{v}_k) + \varphi'(\bar{v}_k)\,(v_k - \bar{v}_k),$$
$$u_{y_k}^2 = [\varphi'(\bar{v}_k)]^2\, u_{v_k}^2. \tag{II.8}$$

It is important to emphasize that the linearized form of the activation function is employed exclusively for the estimation of output uncertainty and not for the prediction itself. Therefore, the uncertainty propagation through the activation reduces to scaling the standard uncertainty of v_k by the magnitude of the local slope of the activation function, evaluated at $\bar{v}_k$. This distinction is essential: if the linearized approximation were to be used during the prediction phase, it would inevitably distort the network's behavior. In particular, the nonlinear activation functions are what allow neural networks to capture complex patterns and relationships in the data. Replacing them with a linear approximation in the prediction stage would suppress these nonlinearities, thereby reducing the expressive power of the model and leading to potentially large errors in the predicted outputs.

Consequently, the linearization should be regarded strictly as a mathematical device for analytical tractability in the context of uncertainty propagation.

The propagated output uncertainty obtained via the LPU must be interpreted at the decision level depending on the learning task. In regression, the pair (μ_y, σ_y) can be read directly as a best estimate and its associated standard uncertainty. In classification, however, the network output is a score (or a class probability), and uncertainty can be incorporated in two distinct ways:

- Decision-level calibration of class confidence (single-number use). The goal is to convert (μ, σ) into a single calibrated measure of class confidence. For binary classification with decision threshold τ, if the uncertain score is modeled as $S \sim \mathcal{N}(\mu, \sigma^2)$, a natural calibrated confidence for class 1 is the probability mass on the class-1 side of the threshold:

$$P(\text{class} = 1 \mid x) \approx P(S > \tau) = 1 - \Phi\left(\frac{\tau - \mu}{\sigma}\right)$$
$$and \tag{II.9}$$
$$P(\text{class} = 0 \mid x) = \Phi\left(\frac{\tau - \mu}{\sigma}\right),$$

where $\Phi(\cdot)$ is the standard normal CDF. This produces a threshold-aware confidence value that directly supports the final decision.

- Second-order uncertainty on class scores/probabilities (distribution-level use). Here, (μ, σ) is used to represent a full distribution over the score/probability, enabling confidence intervals and risk-aware decision rules (e.g., whether the uncertainty band overlaps the decision boundary). This perspective does not collapse uncertainty to a single scalar; instead, it retains the distributional information needed to quantify decision stability under uncertainty.

In this thesis, the proposed probabilistic framework follows the second perspective, because it enables explicit reasoning about classification robustness via confidence intervals and conservative decision criteria.

This framework is specifically designed to incorporate and properly interpret the uncertainty obtained from the LPU, enabling a meaningful translation of raw outputs into probabilistic measures of confidence. By doing so, it allows the evaluation of the reliability of the predicted class labels, offering a principled way to quantify the degree of trust in the classification results. Such a framework is particularly valuable in contexts where decisions depend not only on the predicted class itself but also on the associated confidence level, ensuring that uncertainty is explicitly accounted for in the decision-making process.

In this work, the second-order interpretation is adopted: each predicted output score is modeled as the mean (μ) of a Gaussian distribution, with the propagated standard uncertainty used as its standard deviation (σ). This Gaussian representation should be understood as a local approximation around μ; for bounded probability outputs it is most meaningful when σ is moderate so that the bulk of the probability mass remains within $[0, 1]$.

In multi-class classification, the network output is a vector rather than a single scalar. Let $\mathrm{s}(x) = [s_1(x), \dots, s_C(x)]^\top$ denote the vector of class scores (e.g., logits). Under the first-order LPU, the propagated uncertainty can be expressed at the vector level by approximating $\mathrm{S} = \mathrm{s}(X)$ as a multivariate Gaussian $\mathrm{S} \sim \mathcal{N}(\mu_s, \Sigma_s)$, where $\mu_s = \mathrm{s}(x_0)$ and Σ_s is obtained through local sensitivities (in its most general form, $\Sigma_s \approx \mathrm{J}_s \, \Sigma_x \, \mathrm{J}_s^\top$). This representation naturally accounts for correlations between class scores through the off-diagonal terms of Σ_s, which can be relevant because all class outputs are generated by shared internal features. The nominal predicted class is $\hat{c} = \arg\max_c \mu_{s,c}$. To assess decision stability under uncertainty, a convenient quantity is the margin between the top class and its strongest competitor. Let $c^\star$ be the runner-up class, i.e., $c^\star = \arg\max_{j \neq \hat{c}} \mu_{s,j}$, and define the random margin $\Delta = S_{\hat{c}} - S_{c^\star}$. Under the Gaussian approximation, Δ is also Gaussian with mean $\mu_\Delta = \mu_{s,\hat{c}} - \mu_{s,c^\star}$ and variance $\sigma_\Delta^2 = \sigma_{s,\hat{c}}^2 + \sigma_{s,c^\star}^2 - 2\,\mathrm{Cov}(S_{\hat{c}}, S_{c^\star})$. A conservative risk-of-class-change criterion can then be stated as: if $\mu_\Delta -$

$k\sigma_{\Delta} \leq 0$ (e.g., $k = 2$), the $k\sigma$ uncertainty band intersects the pairwise decision boundary $S_{\hat{c}} = S_{c^{\star}}$, indicating that the predicted class may not be robust to the prescribed input uncertainty. If probabilities are required, $\mathrm{p}(x) = \mathrm{softmax}(\mathrm{s}(x))$ introduces bounded and sum-to-one constraints; in that case, the induced distribution is not Gaussian in probability space. In this thesis, the Gaussian assumption is used locally and primarily for decision-stability reasoning; when σbecomes large, the interpretation must be treated with caution, and Monte Carlo propagation remains a useful benchmark for the full nonlinear mapping.

In the binary case (C = 2), decision stability can be evaluated by studying the interaction between the decision threshold and the Gaussian-distributed output. Particular attention is devoted to regions within one and two standard deviations (1σ and 2σ) from the mean. The 2σ interval is of particular interest, as it encompasses approximately 95% of the samples, which is generally considered sufficient for most practical applications. If a broader confidence interval is required, the proposed methodology can be readily extended. By analyzing the evolution of these confidence intervals as input uncertainty increases, meaningful insights can be obtained regarding the stability, reliability, and resilience of the classification outcomes.

The proposed probabilistic framework captures the expected output and quantifies the confidence associated with each prediction, thereby enabling a more nuanced understanding of the decision-making process of ANNs.

The following contributions are introduced for the first time:

- Conservative worst-case criterion: A strategy is introduced whereby the predicted class is switched whenever the Gaussian distribution at 2σ intersects the decision threshold. This approach facilitates the evaluation of the potential impact of aleatoric uncertainty on classification performance by explicitly quantifying the proportion of samples that may be misclassified due to uncertainty propagation. In doing so, the system's sensitivity to increasing uncertainty is characterized, highlighting thresholds at which predictions may lose reliability.
- Less restrictive criterion: An alternative strategy is proposed in which the intersection of the decision threshold with the narrower 1σ Gaussian interval is considered; in this case, the class prediction is altered if the decision threshold falls within the 1σ region of the distribution. This criterion provides a more balanced perspective, mitigating the risk of overestimating the influence of uncertainty on classification outcomes.

Both approaches demonstrate effectiveness in assessing classification quality under uncertainty, as they enable the derivation of additional information regarding the reliability of results based on the selected confidence interval. In effect, these criteria enrich the interpretation of the

classification outcome by quantifying the model's confidence within the given input uncertainty range, offering decision-makers an added layer of insight.

To determine the outcome of the classification, the intersection between the Gaussian distribution and the binary decision threshold is analyzed, with particular attention to the region defined by the standard deviation σ. In this framework, the decision rule adopted in the first case can be described as conservative: if the Gaussian distribution extended to 2σ crosses the binary threshold, the prediction is flagged as being at risk of class change. This situation indicates a potential overlap between the uncertainty associated with the prediction and the decision boundary of the opposite class. In other words, the model does not rely solely on the most probable class outcome but also accounts for the variability of the estimate. By incorporating the spread of the Gaussian distribution into the decision process, the method provides an additional safeguard against misclassification, especially in cases where the prediction lies close to the threshold. This conservative strategy prioritizes robustness over strict confidence, ensuring that predictions in ambiguous regions are treated with greater caution.

In Figure II.3, the Gaussian distribution, defined over a confidence interval of $\pm 2\sigma$ around the network's predicted value, does not intersect the binary decision threshold. This indicates that the predicted class remains stable despite the presence of uncertainty.

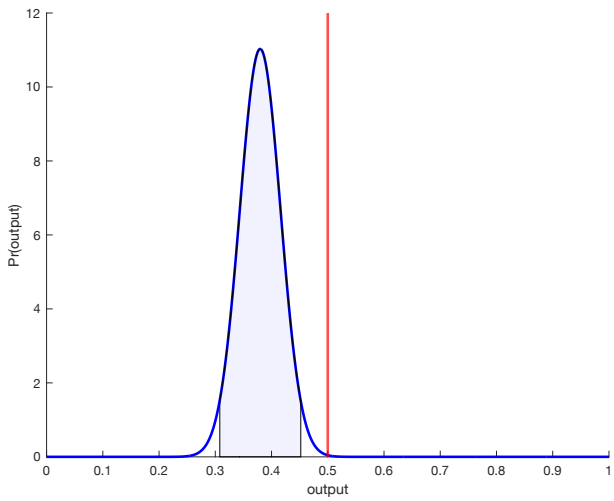

Figure II.3 *2σ thresholding: no intersection between the considered portion of the Gaussian distribution and the binary threshold, resulting in a correct classification.*

In contrast, in Figure II.4, the Gaussian at ±2σ extends beyond the threshold, suggesting that the uncertainty associated with the prediction spans both class regions and therefore introduces ambiguity in the classification. outcome.

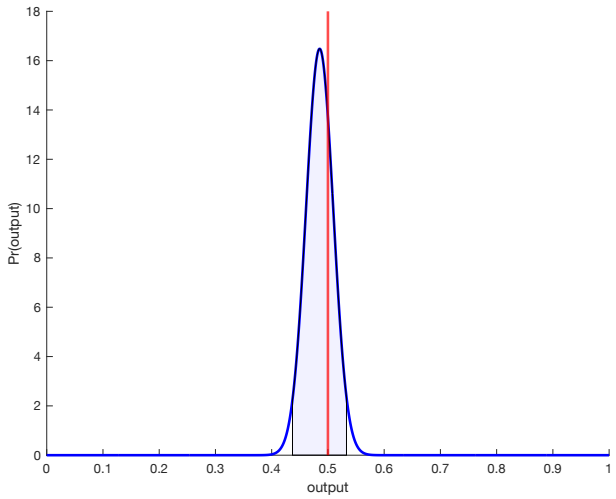

Figure II.4 *2σ thresholding: intersection with the binary threshold, which leads to a potential misclassification.*

To address such cases, a conservative "worst-case" strategy is proposed: whenever the Gaussian distribution at ±2σ intersects the decision threshold, the class is considered as switched. This approach allows for the explicit assessment of the potential impact of aleatory uncertainty on classification performance, by quantifying the fraction of samples that could be misclassified due to uncertainty propagation. The results provide insights into how accuracy and misclassification rates degrade under uncertain conditions. Ideally, such ambiguous cases should be excluded from the classification process and subjected to more rigorous handling, thereby avoiding forced decisions when the level of uncertainty exceeds acceptable limits. The classification outcome is thus determined by examining the position of the Gaussian distribution, centered on the predicted value and relative to the binary decision threshold, with specific focus on the interval defined by ±2 standard deviations.

To estimate the impact of input uncertainty on the classification according to the proposed analysis, an additional study can be carried out by systematically varying the percentage of input uncertainty. By doing so, one can observe how the resulting Gaussian distribution expands accordingly, which in turn makes it possible to evaluate more precisely the influence of

such uncertainty on the final classification outcome. This approach provides deeper insight into the sensitivity of the classification process, highlighting to what extent the model's decision boundaries are robust or vulnerable to different levels of input variability.

For each considered level of input uncertainty, the evaluation procedure involves assessing whether the 2σ confidence interval of the Gaussian distribution intersects the decision threshold. Such an intersection would indicate a potential ambiguity in the assignment of the predicted class, signaling a possible loss of classification reliability. As illustrated in Figures II.5 and II.6, across the entire range of increasing input uncertainty levels examined, the Gaussian distributions at 2σ consistently remain confined to a single side of the decision threshold, without encroaching into the region associated with the opposite class. This consistent separation implies that, even as the uncertainty envelope progressively broadens, the predictions preserve their classification integrity and do not exhibit a risk of class switching. These findings suggest that the system possesses a notable degree of robustness with respect to the propagation of moderate levels of input uncertainty. In particular, the model is able to sustain correct classification decisions despite the deliberate amplification of uncertainty, thereby reinforcing confidence in the stability and reliability of its predictive behavior under perturbed conditions.

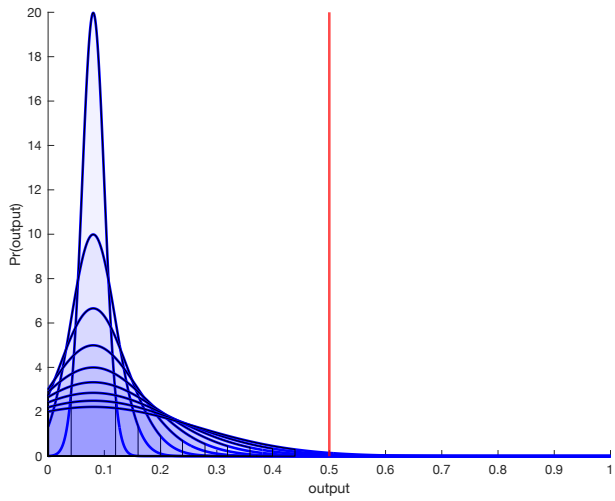

Figure II.5 *Correct binary classification for each level of input uncertainty percentage: class 0.*

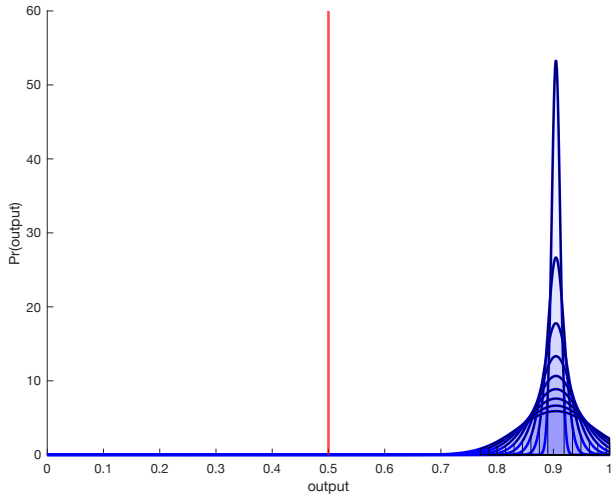

Figure II.6 *Correct binary classification for each level of input uncertainty percentage: class 1.*

In contrast, Figures II.7 and II.8 reveal that for certain levels of input uncertainty, the Gaussian distributions at $\pm 2\sigma$ around the predicted values intersect the binary decision threshold. As the level of uncertainty increases, the Gaussian spread expands proportionally, eventually encroaching into the region associated with the alternative class. This intersection indicates that the propagated uncertainty has reached a magnitude sufficient to introduce ambiguity into the classification outcome, thereby raising the likelihood of a class transition.

Within our framework, such instances are interpreted as potential misclassifications under a conservative, worst-case scenario assumption. The occurrence of these intersections underscores the critical role of uncertainty in the decision-making process: predictions that appear highly reliable under nominal conditions can lose their stability once the associated uncertainty exceeds a critical threshold. This finding highlights the importance of explicitly accounting for uncertainty propagation, as it can fundamentally alter the confidence and reliability of classification results, even in cases where the nominal prediction is strongly biased toward the correct class.

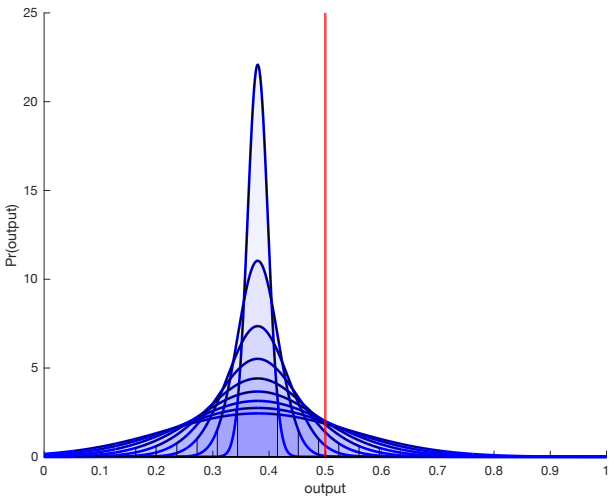

Figure II.7 *Misclassification for certain levels of input uncertainty percentage: class 0.*

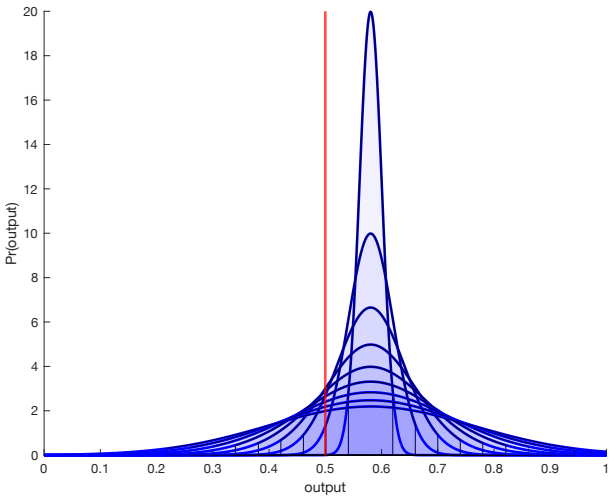

Figure II.8 *Misclassification for certain levels of input uncertainty percentage: class 1.*

Recognizing that the initial hypothesis, namely, assuming a class change whenever the binary decision threshold intersects the Gaussian distribution at the 2σ interval, may be overly conservative and risk inflating the estimated misclassification rate, it is proposed an alternative, less restrictive criterion. Specifically, it is considered the narrower 1σ Gaussian interval, which captures approximately 68.2% of the probability mass around the predicted mean.

This adjustment provides a more balanced and nuanced perspective, as it concentrates on the central region of the probability density where the bulk of predictions are located. By doing so, it mitigates the tendency to overstate the influence of uncertainty on classification outcomes, offering a fairer representation of the system's actual behavior, as illustrated in Figure II.9.

By narrowing the uncertainty envelope, this approach enables a more realistic evaluation of classification stability, effectively distinguishing between marginal overlaps occurring in the distribution's tails, where their practical impact is limited, and meaningful overlaps near the mean, where the risk of genuine misclassification is more substantial. Consequently, the method refines the assessment of uncertainty-induced classification shifts, avoiding undue pessimism while still acknowledging situations where prediction reliability may be compromised.

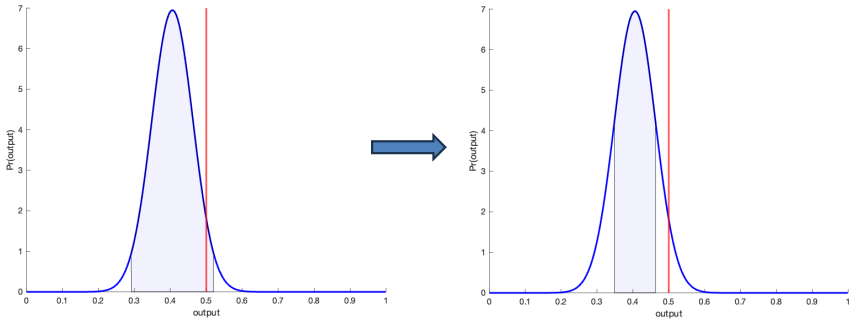

Figure II.9 *From 2σ thresholding to 1σ thresholding.*

II.4 Comparison with Monte Carlo Propagation

After introducing the LPU-based probabilistic framework for classification in Section II.3, this section compares it with Monte Carlo propagation as a numerical reference method. This section clarifies the relationship between the deterministic network prediction evaluated at the nominal input values and the output statistics obtained when input uncertainty is propagated. In

particular, Monte Carlo propagation estimates the distribution of $Y = f_\theta(X)$ induced by the prescribed input distributions of X; therefore, its sample mean approximates $\mathbb{E}[f_\theta(X)]$, which may differ from the deterministic prediction $f_\theta(x_0)$ whenever the network behaves nonlinearly over the range of relevant input variability. This difference is a genuine effect of nonlinear propagation rather than an artefact of Monte Carlo sampling.

In the multi-class case, the same relationship holds by considering a vector output (class-score or probability vector) rather than a scalar. The decision-stability interpretation based on the top-class margin and its uncertainty (Section II.3, Multi-class extension) can likewise be benchmarked against Monte Carlo propagation by estimating the empirical distribution of the corresponding margins.

In the proposed framework, as described in Section II.3, the point prediction is intentionally kept equal to the deterministic model output computed by the fully nonlinear network at the nominal input values, while the associated uncertainty is obtained via a local linearization used only for propagation. Practical Monte Carlo limitations and sampling-related effects in networks with piecewise nonlinear activations (e.g., ReLU dead zones) are discussed in Section II.2.

In regression problems, the decision-level use of propagated uncertainty is more direct than in classification. Let the network output be a continuous quantity $y = f_\theta(x)$. If input uncertainty is prescribed through a distribution for X, Monte Carlo propagation provides a sample set $\{\hat{y}^{(j)}\}_{j=1}^{M}$ from which the output mean $\hat{\mu}_y$ and standard deviation $\hat{\sigma}_y$ can be estimated; these summarize the predictive distribution of the measurand under the given input uncertainty. In the proposed LPU-based approach, the point prediction is the deterministic value $\hat{y}_0 = f_\theta(x_0)$ at the nominal input x_0, while the associated standard uncertainty u_y is obtained analytically via local sensitivity information. Consequently, regression results can be reported in the familiar measurement form $\hat{y}_0 \pm k\,u_y$ (e.g., $k = 1$ for a one-standard-deviation interval or $k = 2$ for an approximate 95% interval under a Gaussian assumption). This provides a transparent and immediately actionable uncertainty statement for continuous outputs, while Monte Carlo remains a useful benchmark when nonlinear effects are strong.

Overall, Monte Carlo propagation, used here as a benchmark, offers high fidelity to nonlinear effects because it propagates full input distributions through the original model, at the cost of many forward evaluations and convergence considerations. In contrast, LPU-based propagation provides a computationally efficient uncertainty estimate through local sensitivity information, but its accuracy depends on the validity of the linear approximation in the operating region.

Chapter III
LPU Assessment in ANNs: From Theory to Real-World Use Cases

The estimation of measurement uncertainty using the LPU in an ANN, according to the outlined concepts, has been applied to different problems ranging from classification to regression. In the case of a regression problem, the uncertainty estimated in this way can be directly interpreted as the uncertainty associated with the prediction itself. In contrast, for classification problems, this parameter is not interpreted as prediction uncertainty in the strict sense but rather is employed to draw conclusions about the confidence of the classification within the proposed probabilistic framework.

In the following, representative case studies are presented, to which this methodology has been successfully applied, highlighting how the approach adapts to the different nature of the problems while maintaining consistency in its theoretical foundation.

III.1 Classification

The problem of classification is particularly compelling and of great significance, as the ability to provide informed predictions is essential in many practical contexts. Below, several application examples are presented.

The first example concerns a comparison between the proposed methodology based on the LPU and Monte Carlo simulation, focusing on a binary classification problem. The aim is to quantify aleatory uncertainty and assess its impact on key performance measures such as accuracy and misclassification rates.

The second example addresses the use of LPU within the range of validity of its linear approximation. This approach is intended to mitigate the issues introduced by the inherent nonlinearity of neural networks, which can

otherwise compromise the reliability of uncertainty estimation when applying the LPU directly.

Finally, the third example deals with a multiclass classification problem in the presence of a nonlinear activation function. Such functions increase both the expressive power and the complexity of the network, thereby making the analytical treatment of uncertainty considerably more challenging.

III.1.1 A Comparison of Analytical vs. Monte Carlo Simulation Approaches to Quantifying Aleatoric Uncertainty

The initial stage of this study consists in the development and training of an ANN for binary classification, as detailed in this section.

III.1.1.2 Training Details

The architecture of the model is presented in Figure III.1. It adopts a simple yet representative design, chosen to ensure clarity in the analysis while still providing sufficient capacity to capture the key patterns within the data. To assess the generality and robustness of the proposed approach, the same neural network architecture is trained on two distinct datasets. Each dataset consists of labeled input–output pairs, enabling supervised learning in which the network is trained to discriminate between the two target classes.

The model is constructed sequentially:

- A first fully connected hidden layer with 32 units and ReLU activation, which helps capture complex patterns in the data.
- A dense hidden layer with 16 neurons and ReLU activation further processes the data.
- An output layer with a single neuron and sigmoid activation, which outputs probabilities suitable for binary classification tasks.

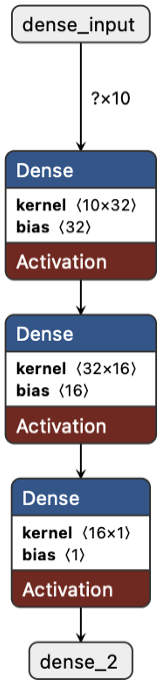

Figure III.1 *Feedforward neural network used for the binary-classification experiments in Chapter III. The model takes a 10-dimensional input feature vector and consists of two fully connected hidden layers (32 and 16 units with ReLU activation) followed by a single sigmoid output neuron. The diagram also reports the corresponding kernel and bias dimensions for each layer.*

The model is compiled with the Adam optimizer at a learning rate of 0.0005, a sensible choice for stable training. It uses binary cross entropy as the loss function and accuracy as the evaluation metric.

To improve training efficiency and prevent overfitting, two callbacks are employed:

- EarlyStopping monitors the validation loss and halts training if no improvement is observed for 30 consecutive epochs while restoring the best weights from the epoch with the lowest validation loss.
- ModelCheckpoint saves the model to a specified path whenever a new best validation loss is achieved, ensuring that the best-performing model is preserved. The model is then trained using the training data over a maximum of 5000 epochs with a batch size of 64, although early stopping typically results in fewer epochs being completed.

Both callbacks are active during training, helping optimize the process and safeguard against overfitting, while verbose logging provides live progress updates. This setup forms a robust training pipeline that balances performance

with protection against overtraining, and it can be further enhanced with techniques such as dropout, batch normalization, or learning rate scheduling for even better results.

III.1.1.3 Datasets Employed

To assess and compare the outcomes of the two approaches, the proposed methods were applied to two datasets that are markedly different in nature, ensuring a comprehensive analysis across diverse data conditions.

- MAGIC Gamma Telescope Dataset (https://www.kaggle.com/datasets/abhinand05/magic-gamma-telescope-dataset).
- Rice Type Classification Dataset (https://www.kaggle.com/datasets/mssmartypants/rice-type-classification).

The training, evaluation, and analysis of the results for the first and second datasets are presented in the next two sections, respectively.

III.1.1.3.1 MAGIC Gamma Telescope Dataset

The MAGIC Gamma Telescope dataset contains 19020 events described by ten continuous image-derived features (Hillas parameters and related descriptors) and a binary label (*gamma* signal vs *hadron* background). The features summarize the geometry and intensity distribution of the Cherenkov shower image (e.g., ellipse length/width, total size, concentration measures, asymmetry, third-order moments, orientation angle, and distance from the camera center), as reported in Table III.4.

From a classification standpoint, this dataset is challenging because the available variables are hand-crafted summaries rather than raw images, and several of them are strongly correlated (e.g., size-concentration-length/width relationships). Moreover, the separation between gamma and hadron events is driven by subtle, distribution-level differences (shape, concentration, and asymmetry), which often produce overlapping class regions in feature space. This overlap can induce ambiguous decision boundaries and makes the posterior confidence sensitive to small perturbations of the inputs-an aspect that is particularly relevant when uncertainty is propagated through the model. In addition, scale disparities among features and non-Gaussian tails/outliers may bias gradient-based training if not properly controlled, motivating the normalization step adopted here.

Table III.4 *MAGIC Gamma Telescope dataset attribute information.*

Attribute name	Description	Type
fLength	major axis of ellipse	continuous
fWidth	minor axis of ellipse	continuous
fSize	10-log of sum of content of all pixels	continuous
fConc	ratio of sum of two highest pixels over size	continuous
fConc1	ratio of highest pixel over size	continuous
fAsym	distance from highest pixel to center, projected onto major axis	continuous
fM3Long	3rd root of third moment along major axis	continuous
fM3Trans	3rd root of third moment along minor axis	continuous
fAlpha	angle of major axis with vector to origin	continuous
fDist	distance from origin to center of ellipse	continuous
class	gamma (signal), hadron (background)	categorical

For model development and evaluation, the dataset was split into 75% for training/validation and 25% for testing. All continuous features were normalized to a common scale to reduce magnitude-driven dominance and improve numerical stability. The best model achieved a validation accuracy of 0.874 with a minimum validation loss of 0.311; the corresponding training and validation curves are reported in Figures III.2 and III.3.

The ANN models and training pipeline were implemented independently (architecture definition, training loop, and evaluation), using standard deep-learning libraries only for basic operations.

Figure III.2 *Training & Validation Accuracy.*

Figure III.3 *Training & Validation Loss.*

Figure III.4 presents the Accuracy and Misclassification rates as functions of the Input Uncertainty levels, offering a clear and comprehensive visual representation of the analysis. This visualization allows for an immediate understanding of how variations in input uncertainty impact the model's performance, highlighting trends and potential thresholds at which the

model's reliability begins to decline. By examining these curves, the robustness of the model under different uncertainty scenarios can be effectively assessed.

As the uncertainty in the input data increases, the corresponding uncertainty in the output also grows, leading more samples to be assigned to incorrect classes. This effect results in a noticeable decrease in test accuracy, accompanied by a rise in the misclassification rate.

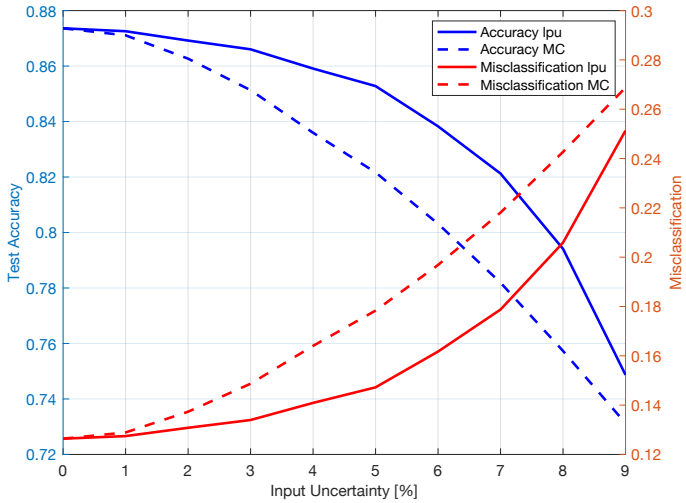

Figure III.4 *Accuracy-Misclassification rates - 2σ analysis.*

The discrepancy between the analytical results, obtained using the law of propagation of uncertainty, and the numerical results from Monte Carlo simulations can likely be attributed to the fundamental differences in how these methods handle non-linearity and input variability. In Monte Carlo simulations, uncertainty is estimated by randomly sampling input values. In the presence of non-linear activation functions such as ReLU, this approach can introduce bias: if a significant portion of sampled inputs falls within the inactive region of the ReLU (i.e., negative inputs), the outputs are forced to zero, artificially shifting the propagated values relative to the expected predictions produced by the classical neural network operating on the original inputs. This effect not only induces errors in the prediction estimates but also inflates the associated uncertainty. Such sampling-induced bias is particularly pronounced in networks where small input variations can lead to disproportionately large changes in the output.

Figures III.5 and III.6 intentionally report *only* the Monte Carlo (MC) results because the plotted quantity is defined as the discrepancy between the MC estimated mean prediction and the nominal deterministic prediction $f(x_0)$. This discrepancy can arise under MC because MC propagates an input

distribution through the full nonlinear mapping and therefore estimates $\mathbb{E}[f_\theta(X)]$, which may differ from $f_\theta(x_0)$. In contrast, the LPU-based approach preserves the nominal prediction by construction and uses linearization only for uncertainty propagation; therefore, the corresponding 'mean-prediction discrepancy' would be identically zero for all samples and all uncertainty levels and plotting it would not add information beyond a flat zero line.

Note that an analogous curve for LPU is not shown because, under the above definition of error, it would coincide with zero everywhere.

In Figures III.5-III.6, the error is defined as $|\ \hat{\mu}_y^{\mathrm{MC}} - f_\theta(x_0)\ |$, where $\hat{\mu}_y^{\mathrm{MC}}$ is the Monte Carlo sample mean of the propagated outputs.

From Figure III.5, it is apparent that the error tends to be higher in the central region of the prediction range, corresponding to values near the binary decision threshold. This region is particularly critical, as predictions close to the decision boundary are more susceptible to class assignment changes, introducing an additional source of error that must be carefully considered. Interestingly, this issue does not arise in the analytical uncertainty propagation approach, where predictions remain stable regardless of variations in the input.

In contrast, at the extremes of the prediction range, where input values are clearly associated with a specific class, the error is significantly lower. This indicates that the network demonstrates greater robustness when classifying inputs that are clearly distinguishable, even in the presence of input variability.

For the LPU approach, the plotted error metric remains identically zero across uncertainty levels, since LPU does not modify the point prediction $f_\theta(x_0)$; for this reason, only MC curves are displayed.

Figure III.6 further emphasizes this behavior. As the input uncertainty increases, the errors estimated via Monte Carlo simulations grow progressively larger, highlighting the contribution of an additional error component that becomes increasingly significant with higher levels of input uncertainty. Once again, this effect is absent in the LPU approach, where predictions maintain consistency despite changes in input conditions. Collectively, these observations underscore the differences in how the Monte Carlo and analytical approaches handle uncertainty, particularly in regions near critical decision thresholds, and highlight the advantage of the LPU method in providing stable and reliable predictions under varying input scenarios.

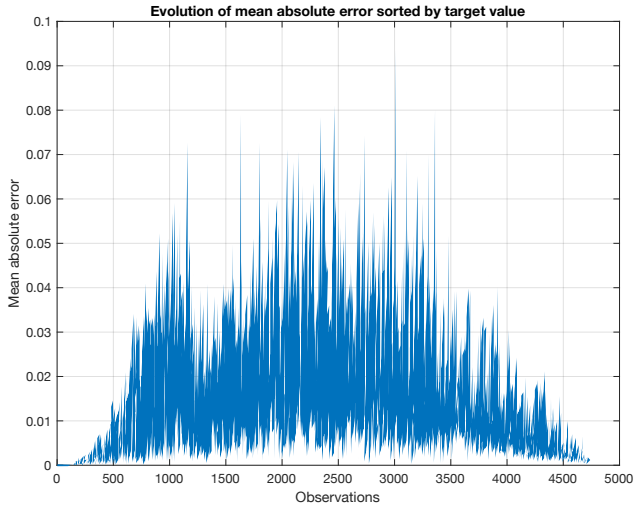

Figure III.5 *Mean absolute error - 2σ analysis.*

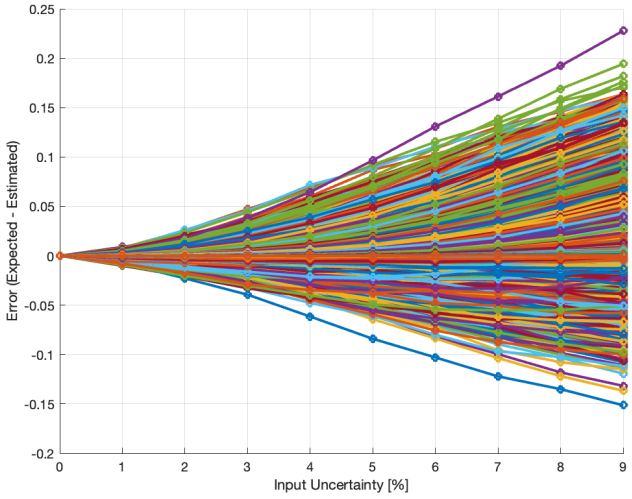

Figure III.6 *Error vs Input Uncertainty - 2σ analysis.*

The propagation of uncertainty approach analytically evaluates uncertainty by linearizing the model around the operating point and computing the sensitivity of the output with respect to the inputs. Importantly, while the uncertainty is estimated using this linearized approximation, the predictions themselves are computed using the full, non-linear neural network. This ensures that the mean predictions remain accurate and are not affected by the approximation errors that could arise from linearization. By adopting this deterministic treatment, the method avoids the pitfalls associated with random

sampling in inactive regions of activation functions, preventing the uncertainty estimation from being skewed by sampling artifacts.

However, the linearization process inherently simplifies the behavior of the network. It may not fully capture complex interactions and higher-order non-linear effects present in highly non-linear systems, which can explain some residual discrepancies observed when compared to Monte Carlo results.

Consequently, the differences between these two methods reflect a fundamental trade-off between computational efficiency and fidelity to complex model dynamics. Monte Carlo simulations can capture intricate non-linear interactions without requiring knowledge of the network's internal structure, weights, or connectivity. However, they are vulnerable to sampling biases, particularly in regions where activation functions suppress variability, leading to potentially inflated or distorted uncertainty estimates. In contrast, the propagation of uncertainty provides a bias-free and computationally efficient estimation of uncertainty, but at the potential cost of overlooking subtler non-linear effects. The discrepancies observed between the two approaches are therefore a natural consequence of these methodological differences, highlighting the complementary strengths and limitations of each technique.

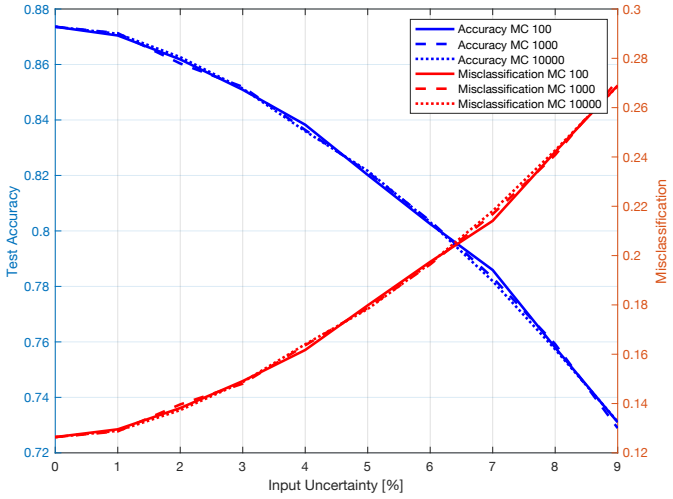

Figure III.7 *Impact of Monte Carlo iterations on Accuracy-Misclassification rates - 2σ analysis.*

Finally, to evaluate the effect of the number of iterations on the outcomes of the Monte Carlo simulation, Figure III.7 illustrates the trends in accuracy and misclassification rate as the number of iterations increases. The results are shown for three different iteration counts: 100, 1000, and 10000. These values also highlight the practical trade-off discussed in Section III.1.1.5: increasing M improves the stability of MC estimates but increases the computational cost

approximately linearly. This comparison provides insight into how the stability and reliability of the simulation results improve with increasing iterations, highlighting the convergence behavior of the Monte Carlo estimates and the diminishing variability in performance metrics as the sample size grows.

The differences discussed above can be more clearly and intuitively appreciated through the confusion matrices shown below, which illustrate the inference results as a function of increasing input uncertainty. The matrix on the left reports the results obtained using the LPU, while the matrix on the right presents the outcomes derived from the Monte Carlo simulation. These visual representations highlight the distinct ways in which the two approaches affect classification performance under varying levels of uncertainty.

As evident from the confusion matrices, increasing input uncertainty (and the corresponding propagation of output uncertainty) leads to a growing number of samples being misclassified. This manifests as a progressive rise in both false positives and false negatives, which, in turn, causes a noticeable degradation in classification accuracy. Such behavior underscores the critical importance of explicitly accounting for the uncertainty inherent in the input data when performing classification tasks.

To complement the visual inspection of the confusion matrices and provide a compact numerical comparison between LPU and MC across all uncertainty levels, the F1-score of the positive (i.e., "true") class is computed for each matrix. With the convention used in Figures III.8-III.17 (rows: True Class, columns: Predicted Class), the entries are interpreted as: $TN = N(false \rightarrow false)$, $FP = N(false \rightarrow true)$, $FN = N(true \rightarrow false)$, and $TP = N(true \rightarrow true)$. The F1-score is then defined as $F1 = \frac{2TP}{2TP+FP+FN}$, which summarizes the trade-off between false negatives and false positives under increasing input uncertainty.

For the 2σ decision criterion (input uncertainty from 0% to 9%), the resulting F1-score "curves" are:

- LPU: [0.906, 0.906, 0.903, 0.901, 0.895, 0.890, 0.880, 0.867, 0.848, 0.816]
- MC: [0.906, 0.904, 0.898, 0.889, 0.876, 0.863, 0.848, 0.829, 0.806, 0.782]

Overall, both methods degrade as uncertainty increases, but MC exhibits a steeper reduction in F1 as the confusion matrices progressively accumulate both FP and FN counts. In contrast, LPU shows a more gradual degradation, indicating higher stability of the class decision under the same uncertainty range.

Moreover, these results demonstrate the necessity of analyzing how different levels of uncertainty in the same dataset influence the network's predictions. Such an analysis not only provides a quantitative measure of the model's reliability and robustness but also offers deeper insight into its operational limits under uncertain conditions. This understanding is particularly valuable in practical applications, where robustness to uncertainty

often represents a decisive factor for the safe and effective deployment of neural networks.

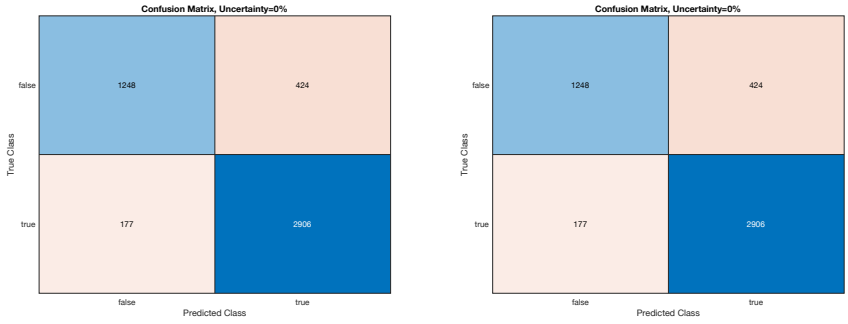

Figure III.8 *LPU & MC confusion matrices for 0% input uncertainty - 2σ analysis.*

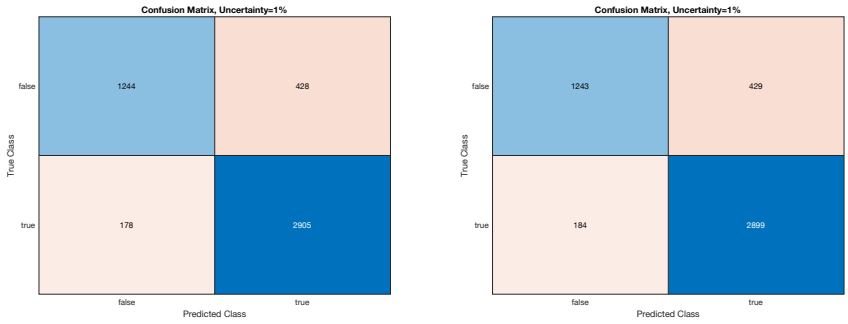

Figure III.9 *LPU & MC confusion matrices for 1% input uncertainty - 2σ analysis.*

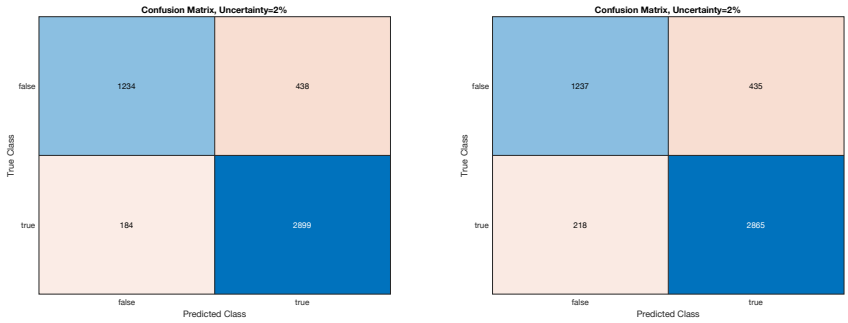

Figure III.10 *LPU & MC confusion matrices for 2% input uncertainty - 2σ analysis.*

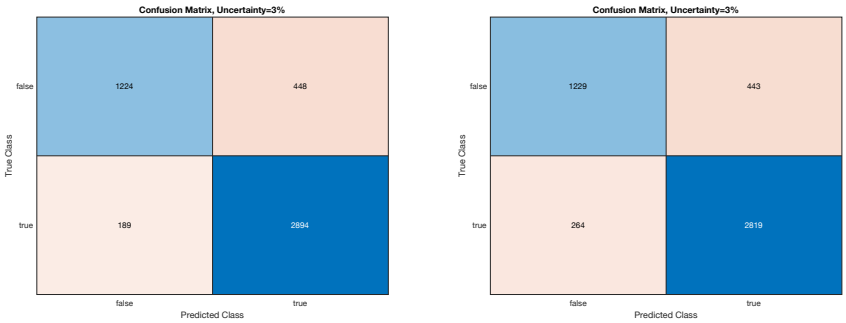

Figure III.11 *LPU & MC confusion matrices for 3% input uncertainty - 2σ analysis.*

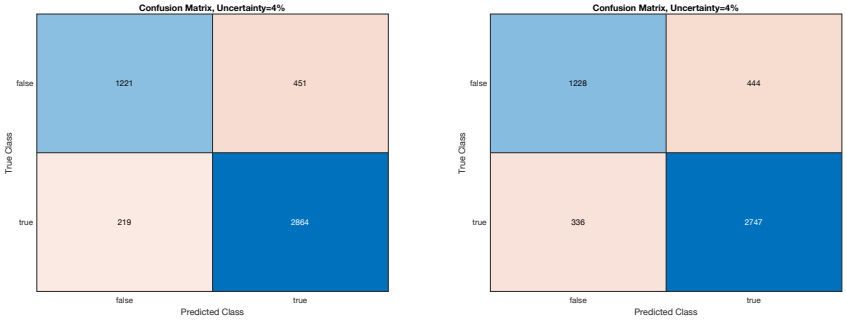

Figure III.12 *LPU & MC confusion matrices for 4% input uncertainty - 2σ analysis.*

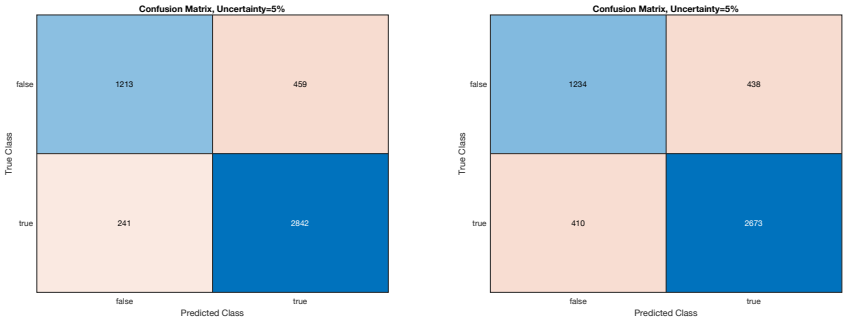

Figure III.13 *LPU & MC confusion matrices for 5% input uncertainty - 2σ analysis.*

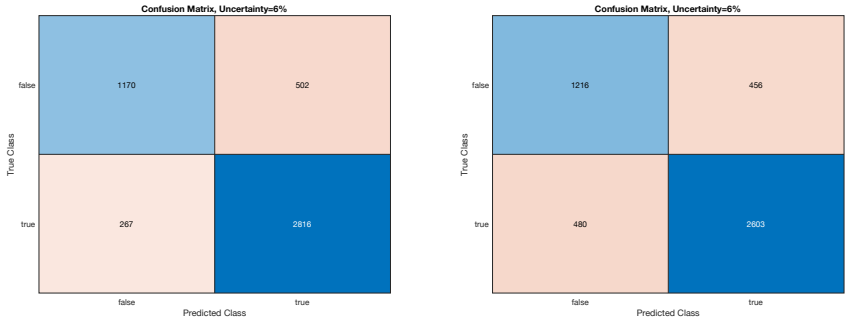

Figure III.14 *LPU & MC confusion matrices for 6% input uncertainty - 2σ analysis.*

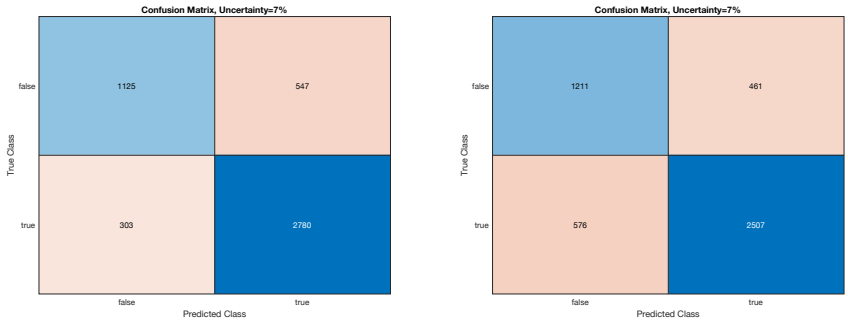

Figure III.15 *LPU & MC confusion matrices for 7% input uncertainty - 2σ analysis.*

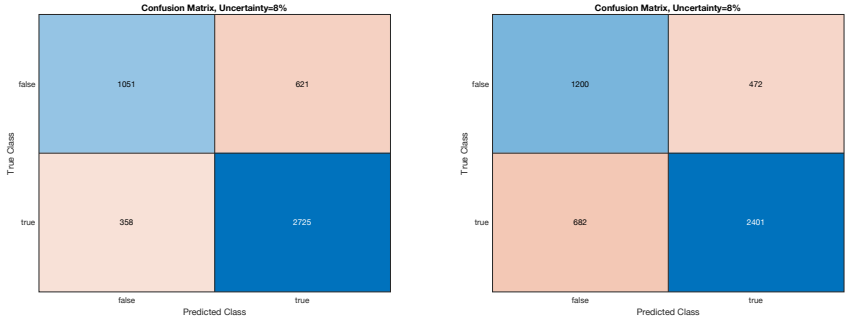

Figure III.16 *LPU & MC confusion matrices for 8% input uncertainty - 2σ analysis.*

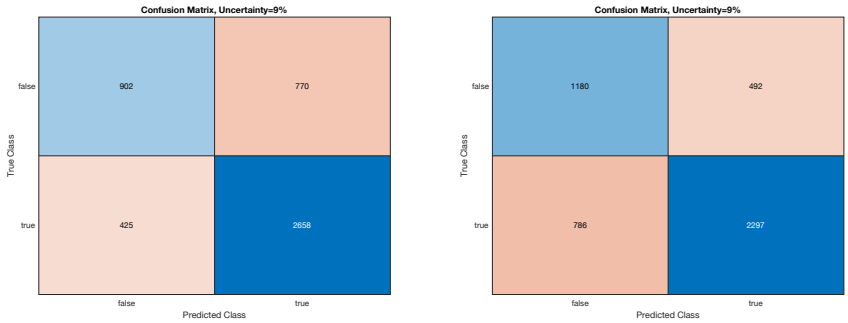

Figure III.17 *LPU & MC confusion matrices for 9% input uncertainty - 2σ analysis.*

Acknowledging that the initial assumption (whereby a class change is considered whenever the binary decision threshold intersects the Gaussian distribution at the 2σ level) may be overly conservative and risk inflating misclassification estimates, the analysis was also extended using a less restrictive criterion.

Specifically, the evaluation was repeated by considering the intersection of the decision threshold with the 1σ region of the Gaussian distribution. In this revised approach, a class change is assumed only if the threshold falls within the 1σ interval, thereby focusing on the core of the probability density where predictions are more concentrated. This adjustment provides a more balanced assessment by reducing the tendency to overestimate the influence of uncertainty on classification outcomes.

By narrowing the uncertainty range under consideration, this method offers a more nuanced evaluation of classification stability, effectively distinguishing between minor overlaps in the distribution tails and more critical overlaps near the mean.

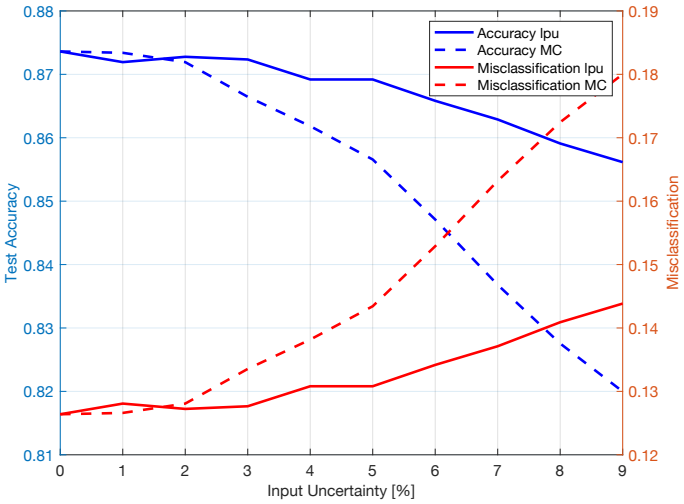

Figure III.18 *Accuracy-Misclassification rates - 1σ analysis.*

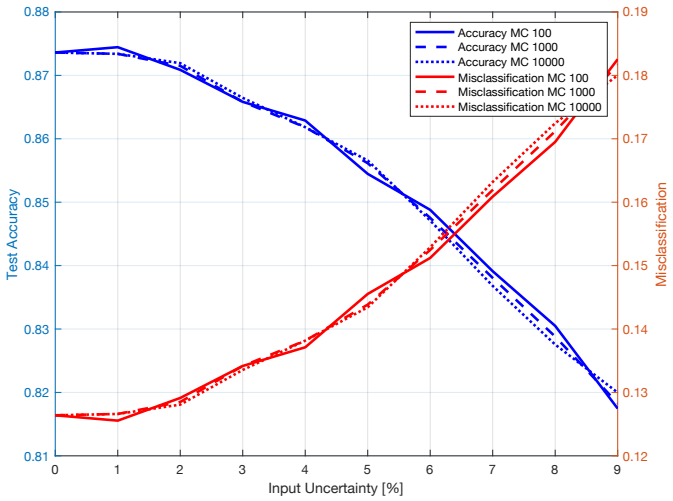

Figure III.19 *Impact of Monte Carlo iterations on Accuracy-Misclassification rates - 1σ analysis.*

Using the same definition as above, the F1-score of the positive ("true") class is reported to numerically summarize how the confusion matrices evolve when the ±1σ overlap criterion is adopted. For input uncertainty from 0% to 9% (Figures III.20-III.29), the F1-score sequences are:

- LPU: [0.906, 0.905, 0.906, 0.906, 0.903, 0.903, 0.901, 0.898, 0.895, 0.893]

- MC: [0.906, 0.906, 0.905, 0.901, 0.897, 0.893, 0.885, 0.876, 0.868, 0.893]

Compared with the 2σ case, the overall F1 variation is smaller, consistent with a less conservative decision rule. Importantly, the confusion matrices show that changes in FP and FN do not necessarily grow monotonically with uncertainty because the decision is driven by discrete threshold-crossing events; therefore, local non-monotonic fluctuations in the summary metric can occur even when uncertainty increases.

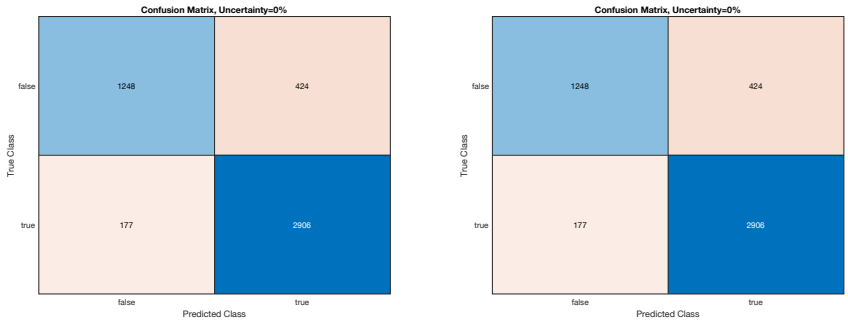

Figure III.20 *LPU & MC confusion matrices for 0% input uncertainty - 1σ analysis.*

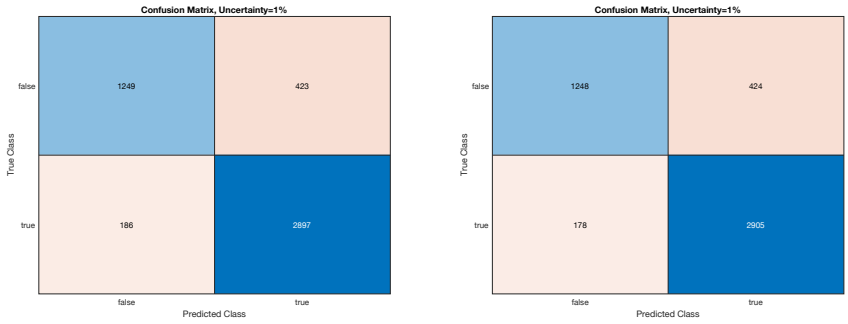

Figure III.21 *LPU & MC confusion matrices for 1% input uncertainty - 1σ analysis.*

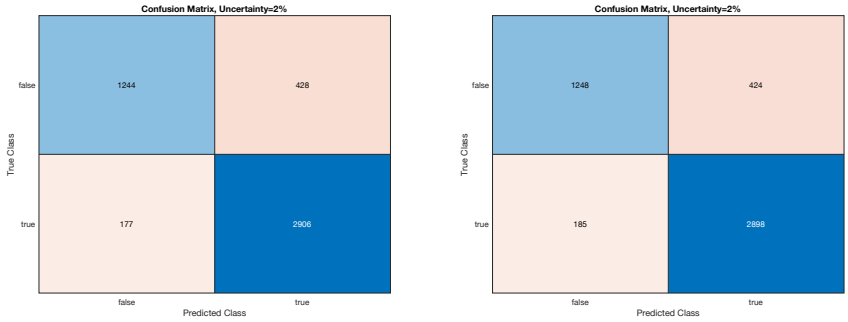

Figure III.22 *LPU & MC confusion matrices for 2% input uncertainty - 1σ analysis.*

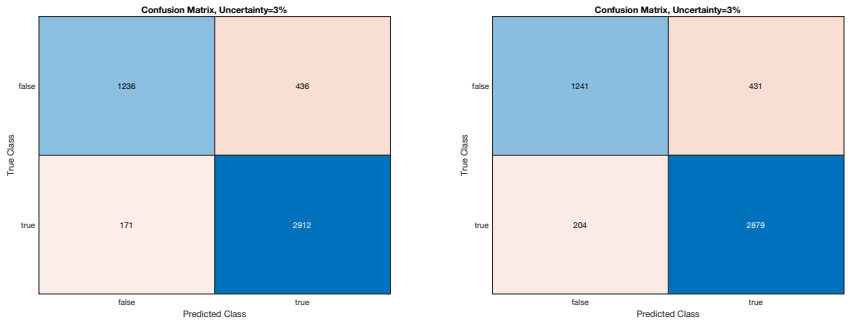

Figure III.23 *LPU & MC confusion matrices for 3% input uncertainty - 1σ analysis.*

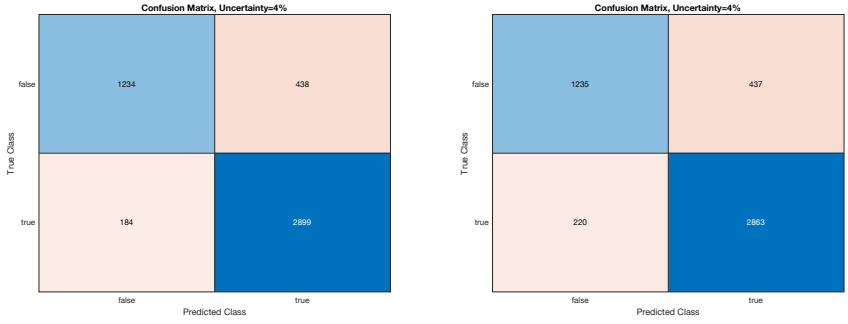

Figure III.24 *LPU & MC confusion matrices for 4% input uncertainty - 1σ analysis.*

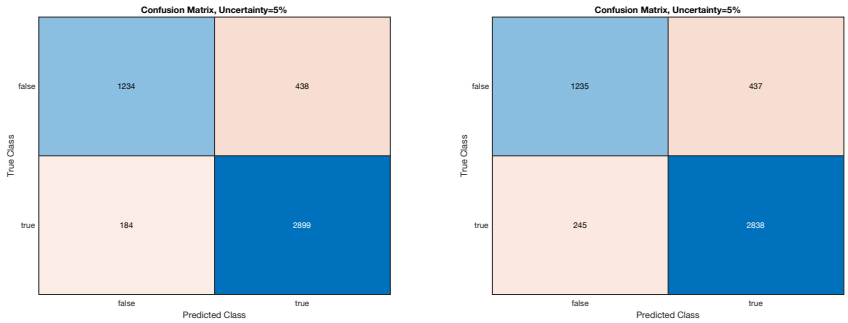

Figure III.25 *LPU & MC confusion matrices for 5% input uncertainty - 1σ analysis.*

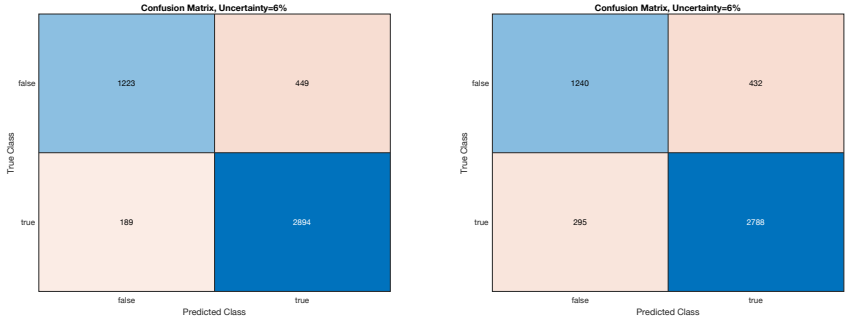

Figure III.26 *LPU & MC confusion matrices for 6% input uncertainty - 1σ analysis.*

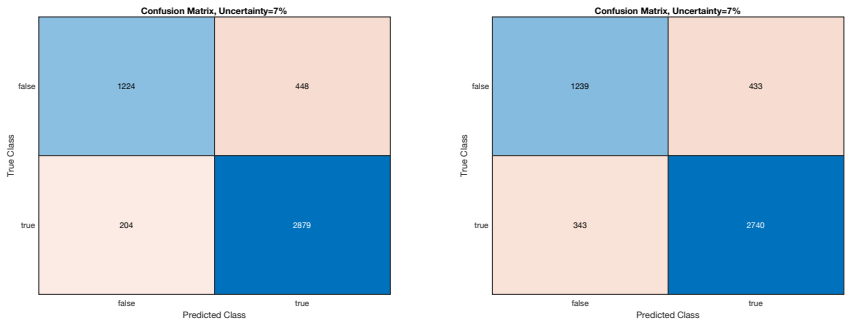

Figure III.27 *LPU & MC confusion matrices for 7% input uncertainty - 1σ analysis.*

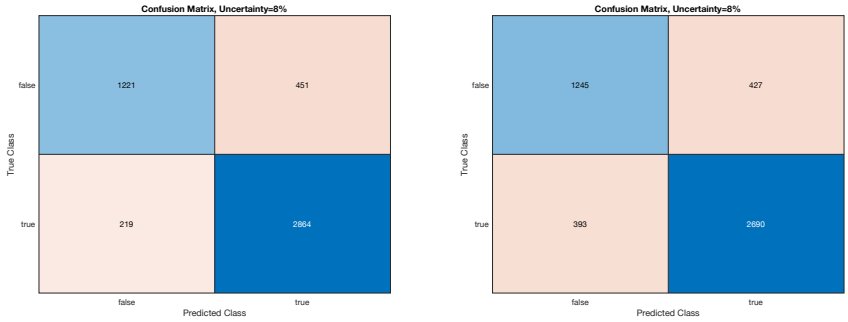

Figure III.28 *LPU & MC confusion matrices for 8% input uncertainty - 1σ analysis.*

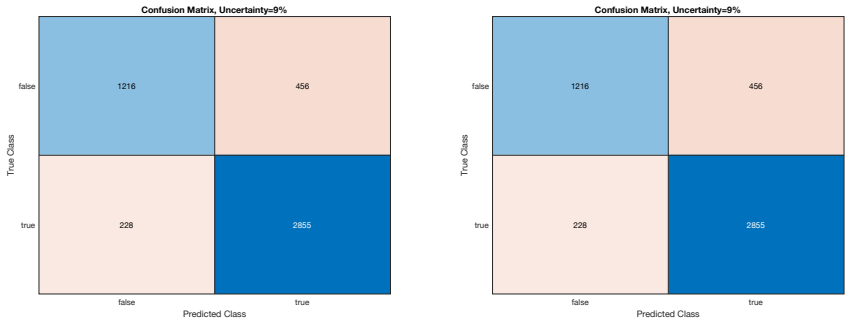

Figure III.29 *LPU & MC confusion matrices for 9% input uncertainty - 1σ analysis.*

III.1.1.3.2 Rice Type Classification Dataset

The same analysis was performed on the second dataset, and the corresponding graphs are presented below. These visualizations provide a direct comparison with the previous results, allowing for an evaluation of whether similar trends and patterns emerge across different datasets. By extending the analysis to an additional dataset, the robustness and generalizability of the observations can be further assessed, offering a broader perspective on the network's behavior under varying data conditions.

The dataset comprises data extracted from two distinct varieties of rice, specifically *Gonen* and *Jasmine*, each contributing unique characteristics and properties that enrich the diversity of the dataset.

The information related to the dataset attributes is shown in Table III.5.

Table III.5 *Rice Type Classification dataset attribute information.*

Attribute name	Type
Area	continuous
MajorAxisLength	continuous
MinorAxisLength	continuous
Eccentricity	continuous
ConvexArea	continuous
EquivDiameter	continuous
Extent	continuous
Perimeter	continuous
Roundness	continuous
AspectRatio	continuous
class	categorical

The dataset comprises 18185 instances in total. For the purposes of model development and evaluation, it was systematically divided into two portions: 75% dedicated to training and validation, and the remaining 25% set aside for testing, thus enabling an objective assessment of generalization capabilities.

The data were subjected to a normalization procedure, which standardized the values of the features to a common scale. This step was essential to mitigate any disparities in feature magnitudes and to enhance the accuracy and reliability of the subsequent analyses.

The optimal performance of the model was observed with a validation accuracy of 0.990 and a corresponding minimum validation loss of 0.032. The training and validation curves are shown in Figures III.30 and III.31, respectively.

Figure III.30 *Training & Validation Accuracy.*

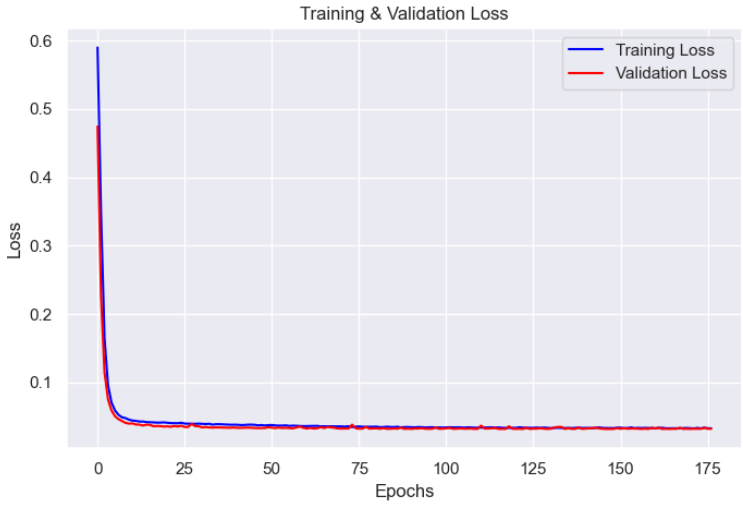

Figure III.31 *Training & Validation Loss.*

The analysis previously conducted for the first dataset at the 2σ confidence level was replicated in this case as well, to enable a direct comparison. Figures III.32 and III.33 show, respectively, the Accuracy-Misclassification rates and the Impact of Monte Carlo iterations on Accuracy-Misclassification rates.

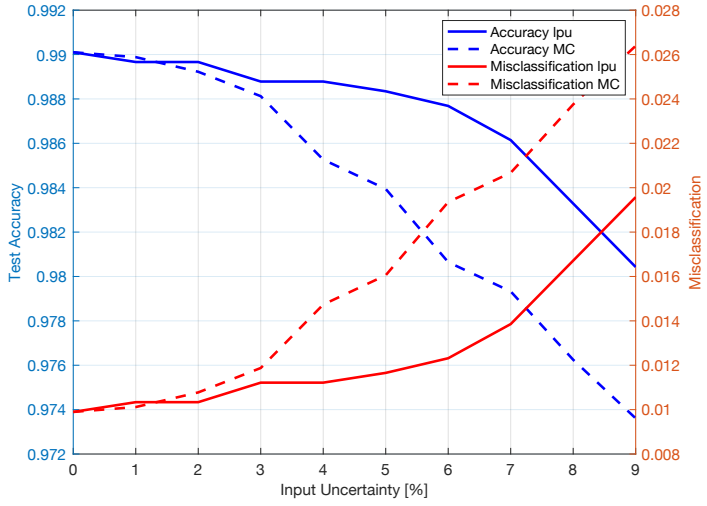

Figure III.32 *Accuracy-Misclassification rates - 2σ analysis.*

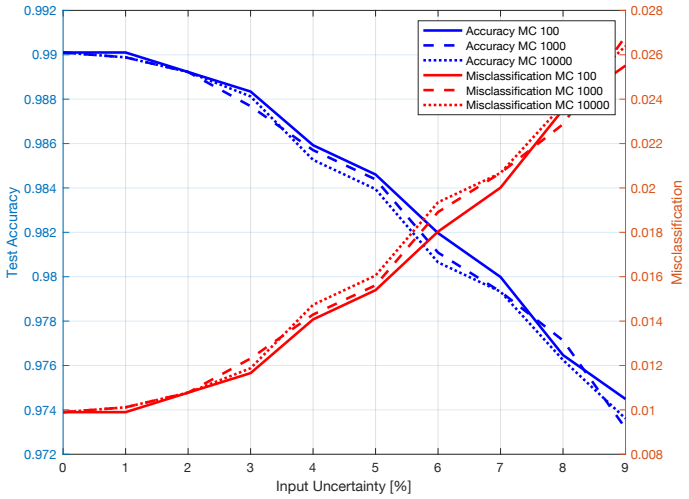

Figure III.33 *Impact of Monte Carlo iterations on Accuracy-Misclassification rates - 2σ analysis.*

Figures III.34-III.35 intentionally report only MC results because the plotted metric is the discrepancy between the MC estimated mean prediction and the nominal deterministic prediction $f_\theta(x_0)$. The analogous quantity for LPU is identically zero by construction and would therefore be uninformative to plot.

Similarly, Figures III.34 and III.35 present the mean absolute error for each prediction in the test set (sorted in ascending order from 0 to 1) and the trend of the errors for each prediction as the input uncertainty increases. This comprehensive and detailed set of figures ensures a thorough and consistent analysis, comparable to the one carried out for the initial dataset.

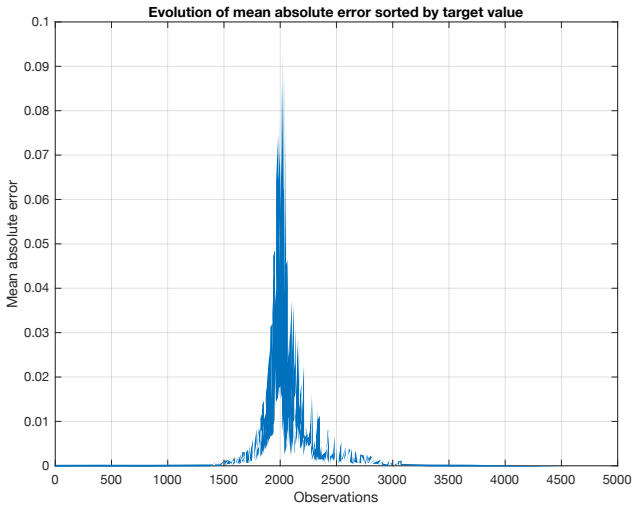

Figure III.34 *Mean absolute error - 2σ analysis.*

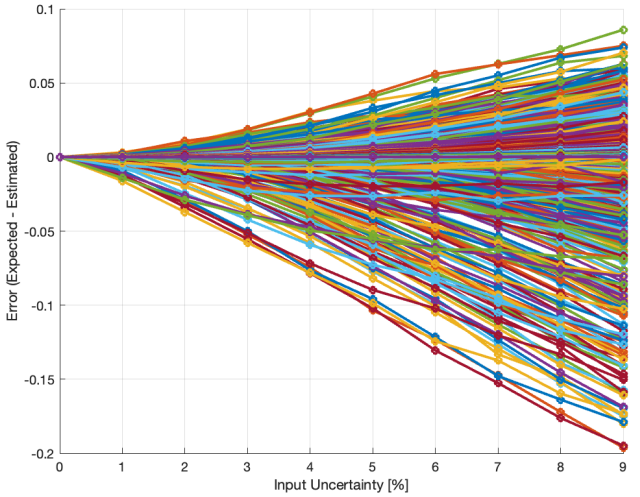

Figure III.35 *Error vs Input Uncertainty - 2σ analysis.*

The confusion matrices are also reported to illustrate the classification performance as the percentage of input uncertainty increases. These matrices provide a clear visual representation of how uncertainty affects the model's ability to correctly classify each class, thereby offering further insight into the

robustness and reliability of each approach under varying levels of input uncertainty.

In addition to the confusion-matrix visualization, the F1-score of the positive ("true") class is computed to provide a compact numerical comparison between LPU and MC as input uncertainty increases. For the 2σ criterion (Figures III.36-III.45, uncertainty from 0% to 9%), the F1-score sequences are:

- LPU: [0.991, 0.991, 0.991, 0.990, 0.990, 0.989, 0.989, 0.987, 0.985, 0.982]
- MC: [0.991, 0.991, 0.990, 0.989, 0.987, 0.985, 0.982, 0.981, 0.978, 0.976]

Even though performance remains high in this use case, the confusion matrices still show a clear trend: MC accumulates errors faster (increasing FP and FN) as uncertainty grows, whereas LPU maintains slightly higher F1 across the full uncertainty range.

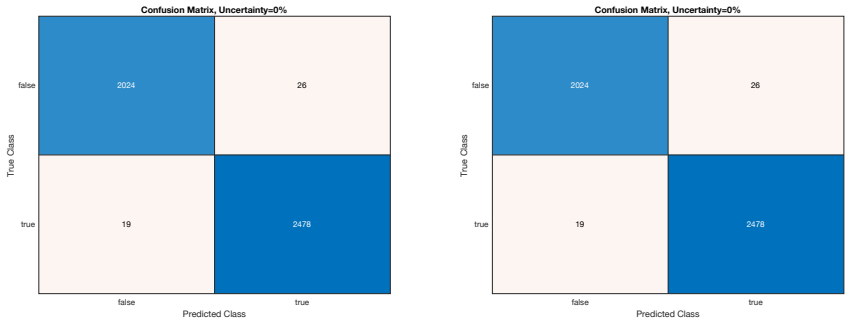

Figure III.36 *LPU & MC confusion matrices for 0% input uncertainty - 2σ analysis.*

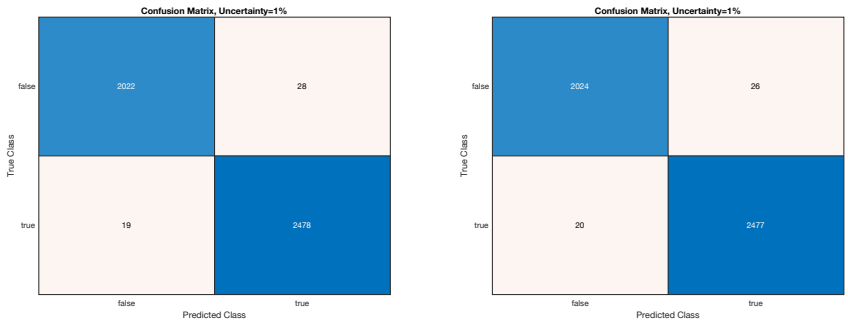

Figure III.37 *LPU & MC confusion matrices for 1% input uncertainty - 2σ analysis.*

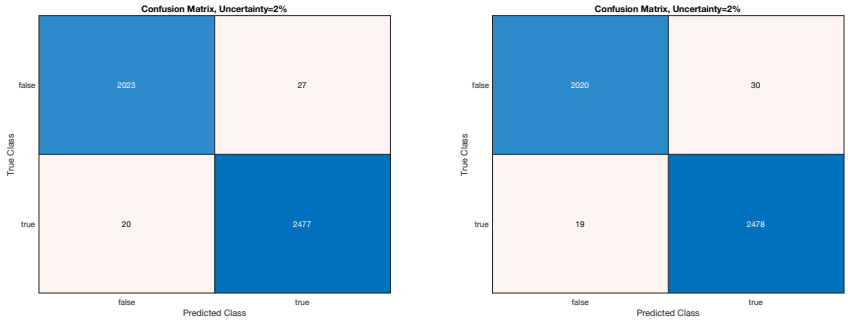

Figure III.38 *LPU & MC confusion matrices for 2% input uncertainty - 2σ analysis.*

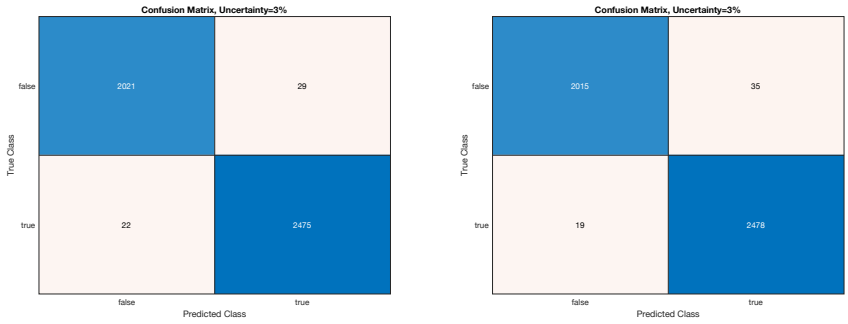

Figure III.39 *LPU & MC confusion matrices for 3% input uncertainty - 2σ analysis.*

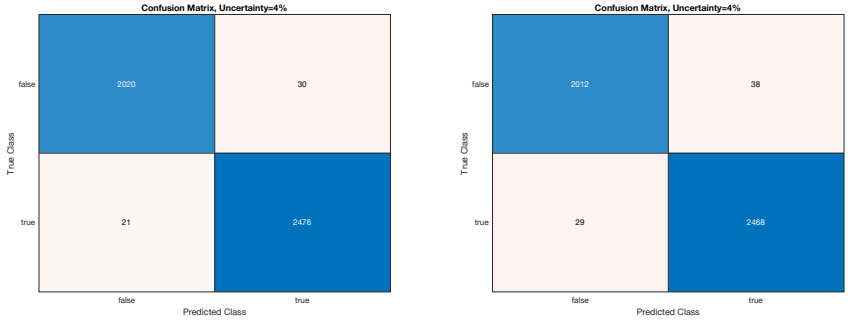

Figure III.40 *LPU & MC confusion matrices for 4% input uncertainty - 2σ analysis.*

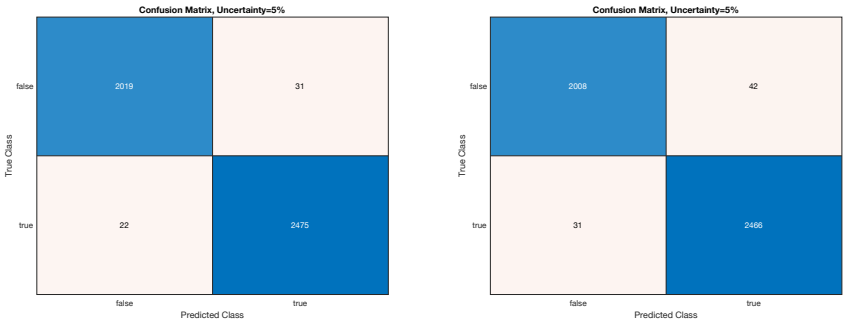

Figure III.41 *LPU & MC confusion matrices for 5% input uncertainty - 2σ analysis.*

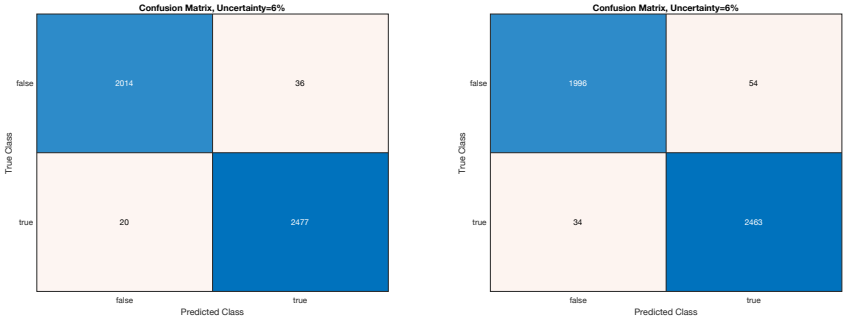

Figure III.42 *LPU & MC confusion matrices for 6% input uncertainty - 2σ analysis.*

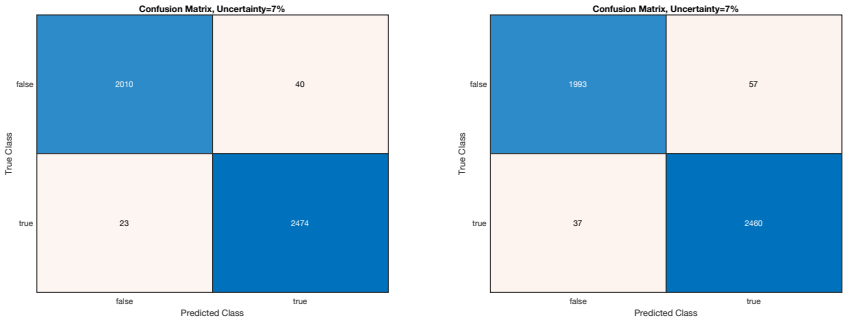

Figure III.43 *LPU & MC confusion matrices for 7% input uncertainty - 2σ analysis.*

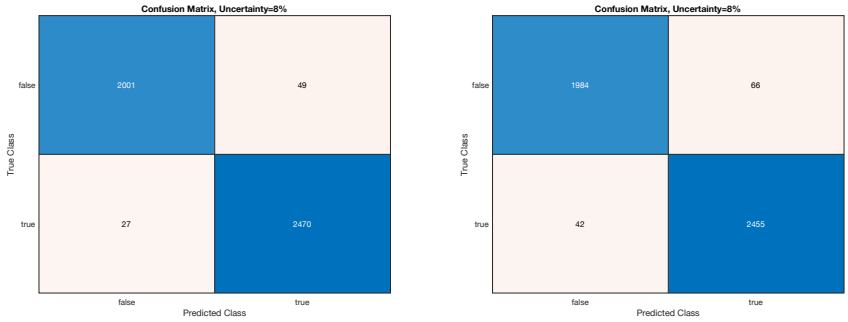

Figure III.44 *LPU & MC confusion matrices for 8% input uncertainty - 2σ analysis.*

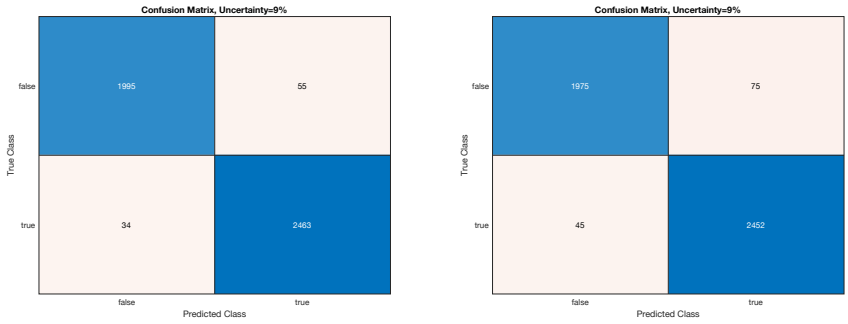

Figure III.45 *LPU & MC confusion matrices for 9% input uncertainty - 2σ analysis.*

Also in this case, to refine the evaluation of classification stability under uncertainty, the analysis was extended beyond the assumption that a class change occurs when the decision threshold intersects the Gaussian distribution at the ±1σ level.

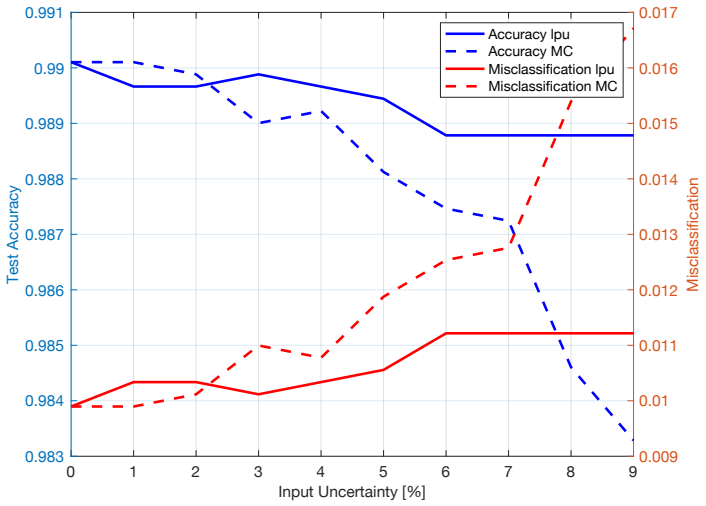

Figure III.46 *Accuracy-Misclassification rates - 1σ analysis.*

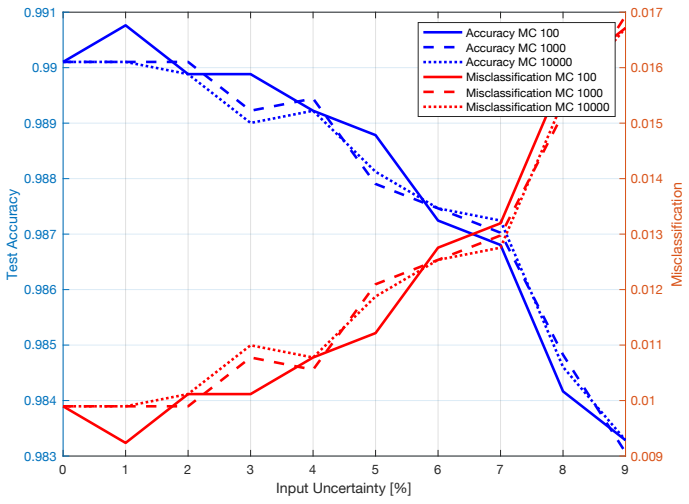

Figure III.47 *Impact of Monte Carlo iterations on Accuracy-Misclassification rates - 1σ analysis.*

To numerically summarize the effect of the ±1σ overlap criterion on the confusion matrices, the F1-score of the positive ("true") class is computed for each uncertainty level. For Figures III.48-III.57 (input uncertainty from 0% to 9%), the resulting F1-score sequences are:

- LPU: [0.991, 0.991, 0.991, 0.991, 0.991, 0.990, 0.990, 0.990, 0.990, 0.990]
- MC: [0.991, 0.991, 0.991, 0.990, 0.990, 0.989, 0.989, 0.988, 0.986, 0.985]

These values confirm, in a compact form, what is visible in the confusion matrices: under the same uncertainty sweep, LPU preserves the decision stability more effectively, while MC shows a more pronounced degradation due to the progressive growth of misclassifications.

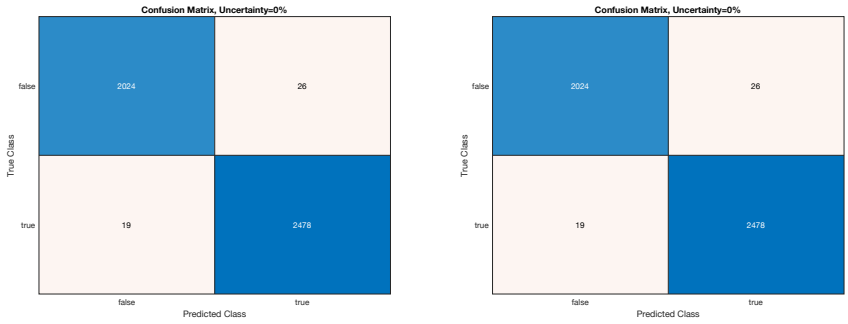

Figure III.48 *LPU & MC confusion matrices for 0% input uncertainty - 1σ analysis.*

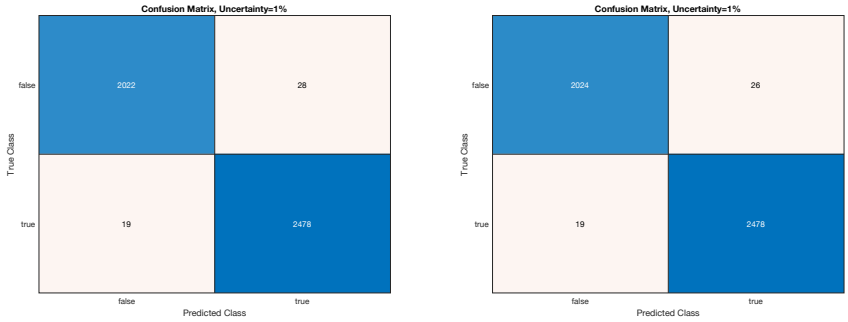

Figure III.49 *LPU & MC confusion matrices for 1% input uncertainty - 1σ analysis.*

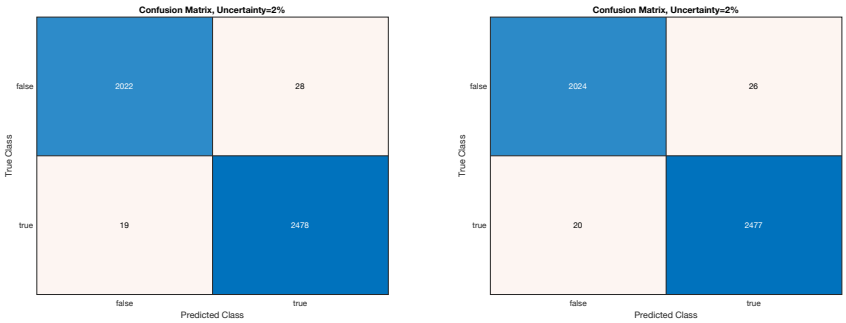

Figure III.50 *LPU & MC confusion matrices for 2% input uncertainty - 1σ analysis.*

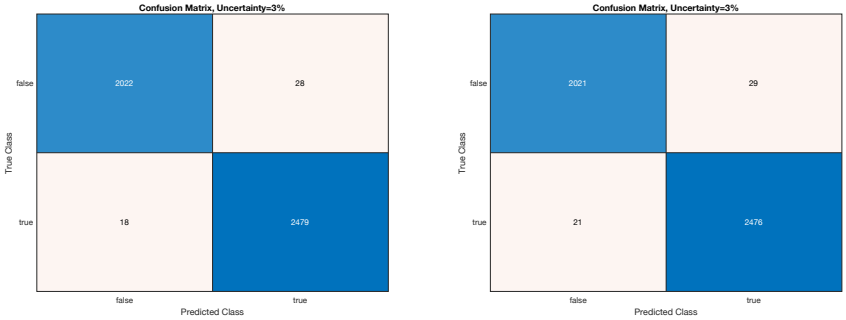

Figure III.51 *LPU & MC confusion matrices for 3% input uncertainty - 1σ analysis.*

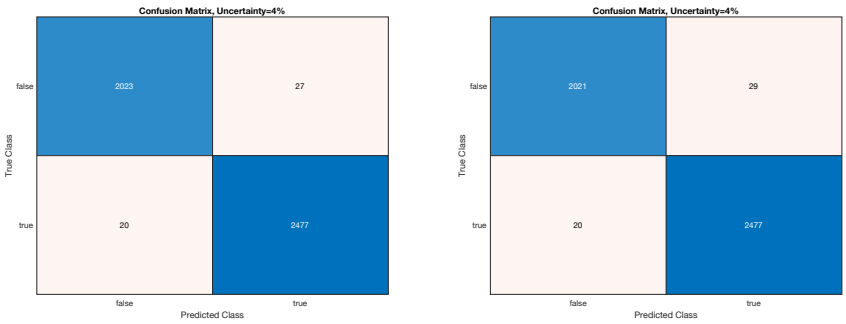

Figure III.52 *LPU & MC confusion matrices for 4% input uncertainty - 1σ analysis.*

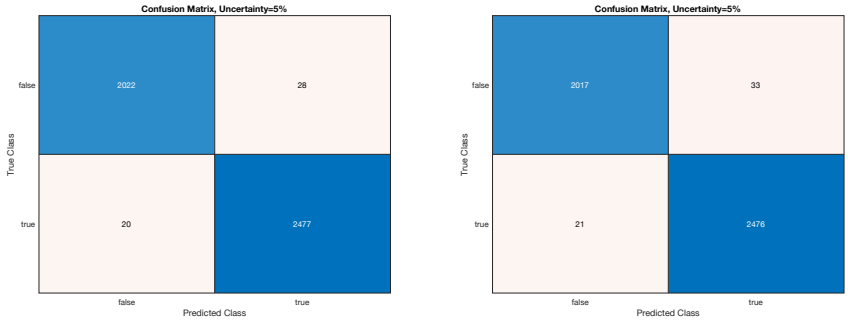

Figure III.53 *LPU & MC confusion matrices for 5% input uncertainty - 1σ analysis.*

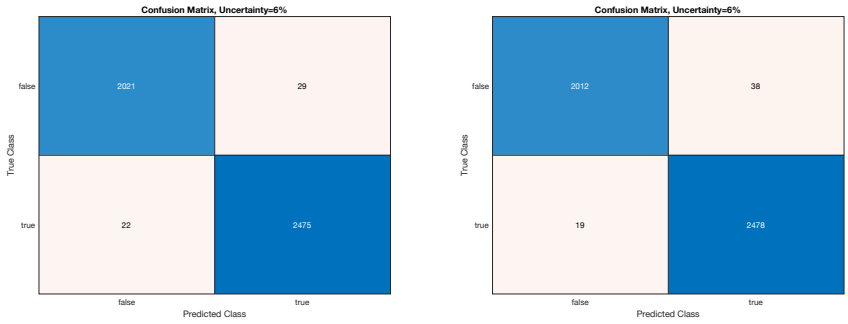

Figure III.54 *LPU & MC confusion matrices for 6% input uncertainty - 1σ analysis.*

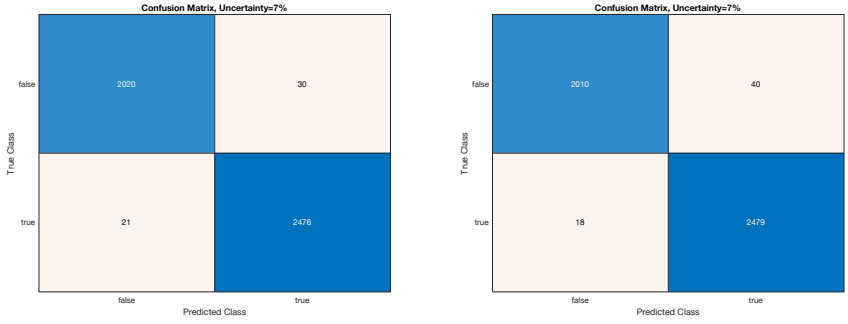

Figure III.55 *LPU & MC confusion matrices for 7% input uncertainty - 1σ analysis.*

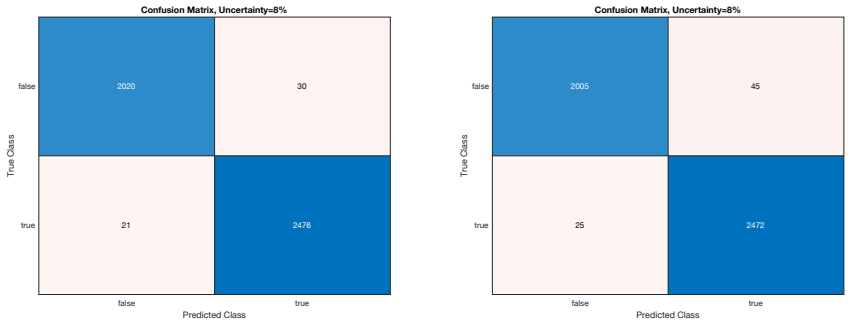

Figure III.56 *LPU & MC confusion matrices for 8% input uncertainty - 1σ analysis.*

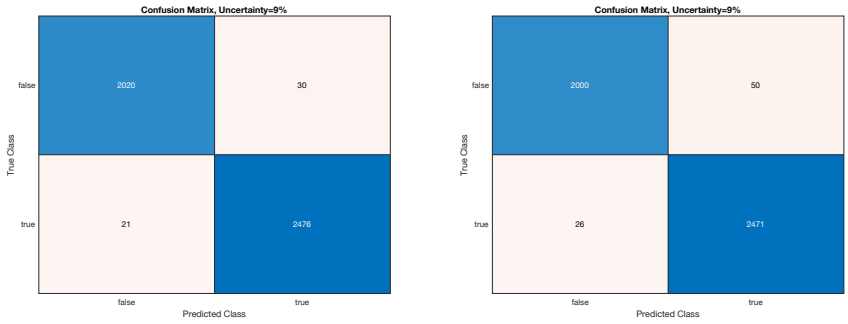

Figure III.57 *LPU & MC confusion matrices for 9% input uncertainty - 1σ analysis.*

III.1.1.4 Findings and Remarks

This study has presented a comprehensive analysis of output uncertainty estimation in neural networks by comparing two complementary methodologies: ISO-GUM-based uncertainty propagation and Monte Carlo simulation. By systematically varying the input uncertainty from 0% to 9% of the nominal value, it was characterized the propagation of uncertainty through the network and its effects on misclassification rates and overall test accuracy.

The probabilistic interpretation of the model outputs, representing each prediction as the mean of a Gaussian distribution with the estimated uncertainty as its standard deviation, has proven effective in providing a deeper understanding of the network's decision-making process. This framework enabled an assessment of classification robustness by examining the interaction between the decision threshold and the probability density function of the outputs. Analysis of the behavior of 1σ and 2σ confidence intervals demonstrated how increasing input uncertainty leads to a broadening

of the output distributions and greater overlap with the decision boundary, thereby increasing the likelihood of misclassification.

The findings derived from the two uncertainty estimation approaches underscore an inherent trade-off between computational efficiency and the ability to capture complex model dynamics. Monte Carlo simulations, while effective in modeling intricate non-linear interactions, are prone to sampling biases, particularly in regions where activation functions dampen variability. In contrast, uncertainty propagation provides an efficient and bias-free analytical estimation but may fail to fully capture higher-order non-linear effects.

Overall, the findings highlight the critical role of input uncertainty in shaping the reliability of artificial neural network predictions. The observed degradation in classification performance with increasing input variability underscores the importance of incorporating uncertainty quantification techniques in the design of neural networks, particularly for safety-critical and high-reliability applications.

III.1.1.5 Computational complexity and runtime comparison

An important advantage of the proposed LPU-based uncertainty propagation over Monte Carlo (MC) propagation is computational efficiency. Let C_{fwd} denote the computational cost of a single forward pass through the trained network and let C_{sens} denote the additional cost required to compute the local sensitivities used for uncertainty propagation (e.g., via analytical derivatives or automatic differentiation). For a scalar output, one LPU evaluation for a single input typically requires one forward pass to obtain the nominal prediction and one sensitivity evaluation, leading to an overall cost $C_{\mathrm{LPU}} \approx C_{\mathrm{fwd}} + C_{\mathrm{sens}}$. In contrast, MC propagation, with M stochastic realizations, requires M forward passes, i.e., $C_{\mathrm{MC}}(M) \approx M\, C_{\mathrm{fwd}}$. Therefore, the expected speed-up factor increases approximately linearly with M: $\text{speed-up}(M) \approx \frac{M\, C_{\mathrm{fwd}}}{C_{\mathrm{fwd}} + C_{\mathrm{sens}}}$. For multi-output networks, the sensitivity-related cost depends on whether a full output Jacobian is required or whether uncertainty is propagated only for a scalar decision statistic (e.g., a class-score margin). In all cases, however, the MC cost remains proportional to M, which makes runtime grow rapidly as M increases.

Absolute wall-clock runtimes depend strongly on the specific MATLAB implementation and the hardware platform. For this reason, this thesis reports computational effort in *normalized units of equivalent forward passes*. Under this convention, one LPU evaluation corresponds to $1 + \rho$ forward passes, where $\rho = C_{\mathrm{sens}}/C_{\mathrm{fwd}}$, whereas one MC evaluation corresponds to M forward passes. This representation makes the computational advantage explicit and directly comparable across platforms. All experiments were executed in MATLAB on an Apple MacBook Pro (14-inch, M1 Pro).

III.1.2 Incorporating Measurement Uncertainty for Enhanced Reliability in Binary Classification Neural Networks

From the analysis of the previous case, and from the comparison between LPU and the Monte Carlo simulation, it emerged that the nonlinearities of the activation functions can introduce distortions and biases both in the network output and in the estimation of the associated uncertainty. In the Monte Carlo case, these nonlinear effects may lead to systematic deviations in the statistical characterization of the results. In the case of the LPU, instead, they significantly increase the analytical complexity of the formulation, making it difficult, especially in a first-order linear approximation, to fully capture the contribution introduced by higher-order terms.

An application of the LPU is presented below, restricted to the operating range in which the activation functions can be considered approximately linear. This assumption allows the analytical formulation to remain tractable while minimizing the distortions typically introduced by nonlinear effects.

III.1.2.1 Dataset and ANN Architecture

The LPU was subsequently applied to an Artificial Neural Network designed for human activity recognition, to highlight its effectiveness in estimating aleatoric uncertainty. The selected classifier, a Multilayer Perceptron, aims to predict an individual's behavior from sensor-based input features. With the rapid progress in healthcare technologies and the growing importance of modeling biological behavior, human activity recognition is becoming increasingly widespread. Much of the current research in this field relies on the classification of sensory signals collected from one or more accelerometers. Owing to their small size, low power requirements, low cost, and unobtrusiveness, accelerometers have become the most widely adopted sensors for capturing motion-related data directly associated with human body gestures.

The dataset used in this study is the MHEALTH dataset (Banos et al., 2014), which contains recordings of body motion and vital signs collected from ten volunteers with diverse demographic profiles. Each participant performed twelve distinct physical activities: standing still, sitting and relaxing, lying down, walking, climbing stairs, bending the waist forward, frontal elevation of the arms, knee bending (crouching), cycling, jogging, running, and performing front and back jumps. Data acquisition was carried out using Shimmer2 wearable sensors (Burns et al., 2010) positioned on the participants' chest and wrists. The use of multiple sensors enabled the measurement of motion across different body segments, capturing

acceleration, rotational velocity, and magnetic field orientation, thereby providing a more comprehensive representation of body dynamics. All modalities were sampled at 50 Hz, a frequency considered adequate for monitoring human activities. Owing to the diversity of movements (e.g., frontal elevation of arms vs. knee bending), activity intensities (e.g., cycling vs. sitting and resting), and execution speeds (e.g., running vs. standing still), the dataset is well-suited to represent everyday activities and generalize across different real-life scenarios.

A simplified version was curated using this dataset as a base, concentrating primarily on two classes: standing (label: '1') and moving (resulting from the grouping of all the non-static activities, label: '0'). This subset's classification is based solely on data collected from the two triaxial accelerometers on the right wrist (a_{rx}, a_{ry}, a_{rz}) and the left ankle (a_{lx}, a_{ly}, a_{lz}).

The classification model is built on examining 70% of the training data with known class labels, while the other 20% of the dataset is the validation subset. As testing samples, the model was run on 10% of all actual cases. The training dataset's min, max, and standard deviation were used in the normalization procedure.

$$x_{i,norm} = \frac{x_i - x_{min}}{x_{max} - x_{min}}. \quad \text{(III.1)}$$

To execute uncertainty propagation inside the neural network, it is required to understand the uncertainty associated with the network inputs. In this use case, the input standard uncertainty was derived from the accelerometer datasheet sensitivity specification. Datasheet values are typically reported under controlled laboratory conditions and mainly reflect the nominal sensor sensitivity and ideal noise behavior. In real wearable recordings, additional effects can increase the effective measurement variability, including sensor bias and drift, temperature dependence, axis misalignment, strap/attachment micro-motions, quantization and digital filtering, and motion artefacts. Since no dedicated metrological calibration of the specific acquisition setup was performed, a conservative multiplicative safety factor of 20 was applied to the datasheet-based uncertainty to avoid underestimating the true measurement uncertainty. This factor is used to obtain a plausible order of magnitude for uncertainty propagation and to study trends, rather than to claim a calibrated absolute uncertainty level.

$$u_{x_{i,norm}} = \frac{u_{x_i}}{x_{max} - x_{min}}. \quad \text{(III.2)}$$

Equation (III.2) applies the same min-max normalization used for the input features to the corresponding standard uncertainties. This is necessary because both Monte Carlo sampling and the LPU sensitivity coefficients are computed in the normalized input space. If an input is normalized as $x'_{\mathrm{norm}} = (x' -$

$x_{\min})/(x_{\max} - x_{\min})$, then the associated standard uncertainty transforms according to the same affine scaling, yielding $u(x'_{\text{norm}}) = u(x')/(x_{\max} - x_{\min})$. Normalizing the uncertainties therefore ensures unit consistency between the inputs used by the network and the uncertainty values propagated through the model.

As shown in Figure III.58, the neural network used in this study is an MLP with three layers:

- The first layer is made up of 5 neurons with ReLU activation function.
- The hidden layer is made up of 3 neurons with ReLU activation function.
- The output layer comprises a single neuron with a sigmoid activation function.

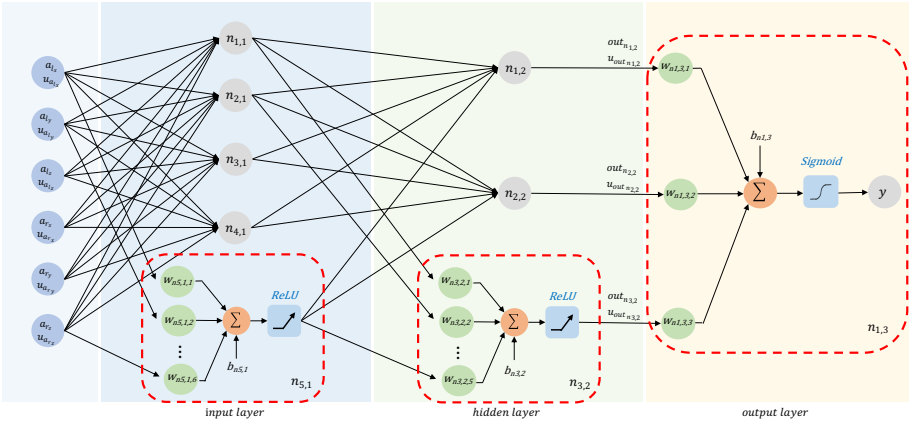

Figure III.58 *Multilayer Perceptron structure employed, with an explicit representation only of the last neuron for each level.*

III.1.2.2 Linearization correctness verification

For the hidden layers, downstream of the ReLU activation functions, the output of the neurons looks as follows:

$$\varphi(v_k) = \begin{cases} v_k, & v_k > 0 \\ 0, & v_k \leq 0 \end{cases}, \tag{III.3}$$

with

$$\varphi'(v_k) = \begin{cases} 1, & v_k > 0 \\ 0, & v_k \leq 0 \end{cases}. \tag{III.4}$$

For the output layer, instead, downstream of the sigmoid activation function, the output of the neuron looks like:

$$\varphi(\upsilon_k) = \frac{1}{1+ e^{-\upsilon_k}}. \quad \text{(III.5)}$$

Since the sigmoid has a non-linear behavior, according to the ISO-GUM, this activation function must be linearized using a first-order Taylor series approximation around υ_0, which is the input value to the activation function obtained each time based on the specific inference parameters:

$$\varphi_{Taylor}(\upsilon_k) = \varphi(\upsilon_0) + \varphi'^{(\upsilon_0)}(\upsilon_k - \upsilon_0), \quad \text{(III.6)}$$

with

$$\varphi'^{(\upsilon_0)} = \frac{e^{-\upsilon_0}}{2e^{-\upsilon_0} + e^{-2\upsilon_0} + 1}. \quad \text{(III.7)}$$

For calculating uncertainty, in this case, the expression became:

$$u_{y_k}{}^2 = \varphi_{Taylor}'(\upsilon_k)^2 \, u_{\upsilon_k}{}^2 = \varphi'(\upsilon_0)^2 \, u_{\upsilon_k}{}^2. \quad \text{(III.8)}$$

The correctness of the assumption of linearization of the sigmoid through first-order truncated Taylor series expansion was verified experimentally. In calculating the final uncertainty, the multiplicative term is represented precisely by the square of the first derivative of the sigmoid function evaluated at the point υ_0. Analysis of the first derivative of the sigmoid indicates that the maximum slope obtainable from this function and located for the function input equal to 0 is 0.25, limiting the maximum sensitivity coefficient used in the uncertainty propagation law.

The assumption of normal distribution at the output of each layer is only valid if the activation function in question is linear; otherwise (as in the case of the sigmoid), one must use the linearized version of the latter (first-order truncated Taylor series expanded version) and assess the goodness of the linearization (Figure III.59). To prove experimentally that around the zero-entry point of the sigmoid activation function the linearization hypothesis is valid, it was decided to perform a linear regression around that point. This regression corresponds to a Maclaurin series expansion of the sigmoid function. This assumption is obviously not valid for the entire domain of the sigmoid function, but it is valid for the values around 0, which are also the critical points in the class choice process.

For these reasons, an evaluation was conducted to determine the maximum interval around the null point that could be reliably approximated by a linear

model. This was achieved by computing the coefficient of determination (R^2) for the linear fit across multiple intervals centered on that point.

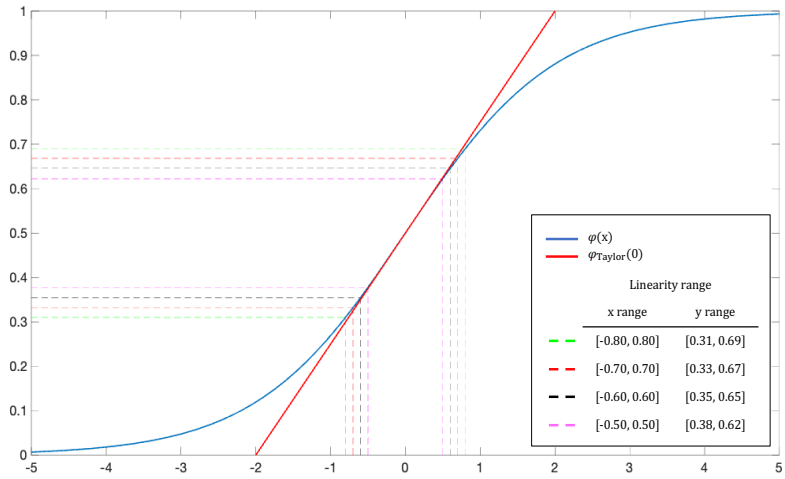

Figure III.59 *Representation of the sigmoid activation function and its linearization as a zero-centered Taylor series expansion. The dotted lines indicate regions where the function and its approximation are nearly equivalent.*

III.1.2.3 Results

Subsequent analyses were conducted on the first linearity range (Figure III.60) because this proves to have a reasonable degree of approximation and, being larger, encompasses a more significant number of samples (49 of the 5530 in inference dataset).

Outside this zone, in theory, it is no longer possible to use the LPU directly, but this has no significant implications as the samples in question are very far from the binary threshold, so there is no need to analyze them in terms of measurement uncertainty to estimate their class.

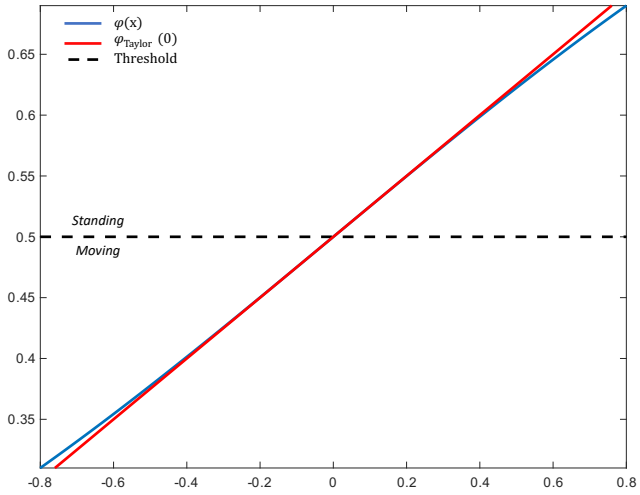

Figure III.60 *Focus of the activation function in the class decision threshold value.*

The probabilistic framework was used to obtain results on the characteristics of the samples falling within the linearity range (Table III.6). Having fixed a specific value of α and thus of the corresponding confidence level CL, it is possible to determine the degree of confidence at which the prediction is provided. Samples whose distribution falls in the zone of noncompliance (the threshold is exceeded) cannot be classified, so it is not possible to provide information on their class. As risk increases, samples can be classified with a higher confidence level, but as this is done, the number of unclassified samples also increases. By reducing the risk, on the other hand, more and more samples can be classified, but with a lower confidence level (Table III.7).

Nevertheless, even with the incorporation of measurement uncertainty, which may result in some test samples not being assigned to their class, the improvement lies in acknowledging that potential fluctuations in the input data, attributed to the sensor used during acquisition, have been considered, ensuring a comprehensive and more appropriate analysis of the problem under consideration.

Table III.6 *Linear regression and linearity ranges.*

Linearity range		R^2	Samples in range
x range	***y range***		
[-0.80, 0.80]	[0.31, 0.69]	0.998	49
[-0.70, 0.70]	[0.33, 0.67]	0.999	44
[-0.60, 0.60]	[0.35, 0.65]	0.999	36
[-0.50, 0.50]	[0.38, 0.62]	0.999	31

Table III.7 *Predictions and corresponding confidence levels.*

α **CL**	0.00024 99.98%	0.0013 99.87%	0.01 99.00%	0.05 95.00%	0.10 90.00%	0.20 80.00%
Not Classified	29	27	20	15	12	8
Classified	20	22	29	34	37	41

From the analysis, it can be observed that within the considered linearity range, where the activation function is accurately approximated, it is possible to draw reliable conclusions about the confidence of the predictions that fall within this interval. This, in turn, enhances the robustness and reliability of the classifier, effectively mitigating the issues introduced by nonlinearity.

III.1.3 Incorporating Measurement Uncertainty for Enhanced Reliability in Multiclass Classification Neural Networks

In the case of a multiclass classifier, the situation becomes even more complex. In fact, it is no longer possible to analyze the intersection between a binary threshold and the output Gaussian distribution of interest, as is typically done in the binary case. Instead, the approach must be modified: one needs to study the intersections among the multiple Gaussian output distributions, one

for each class. This means that the decision boundaries are now determined by the regions where different Gaussian curves overlap, and the classification problem reduces to identifying which class distribution dominates in each region. Consequently, the analysis requires comparing all possible pairs of Gaussian outputs, rather than simply evaluating a single threshold, which significantly increases both the mathematical and computational complexity of the problem.

Having fixed a given confidence level CL, it is possible to determine the degree of confidence that the interval contains the parameter of interest (in this case, the tail of the directly neighboring Gaussian). The corresponding area subtended by the right tail of the left Gaussian is then determined. It is then checked whether it is wholly contained in the area subtended by the right Gaussian (similar speech for the left tail of the right Gaussian and the area of the left Gaussian); in this way, it is possible to assign (Figure III.61) or not (Figure III.62) the class of interest to the samples analyzed with the chosen CL. It is not possible to assign a class to the predictions such that the right (left) tail of the left (right) Gaussian is wholly contained within the area of the right (left) Gaussian defined by the established CL.

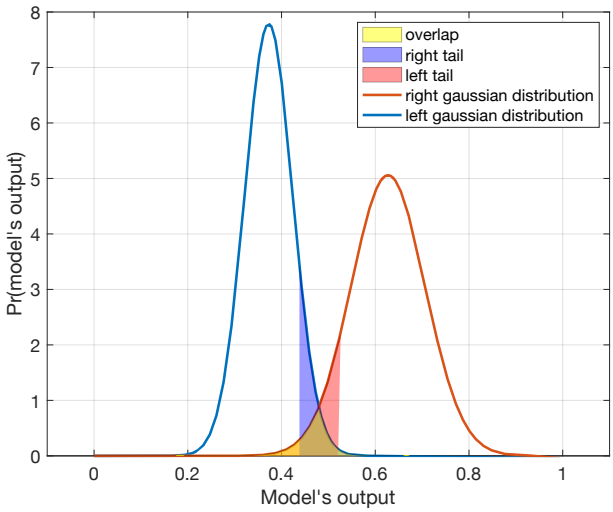

Figure III.61 *Model classification with CL confidence level.*

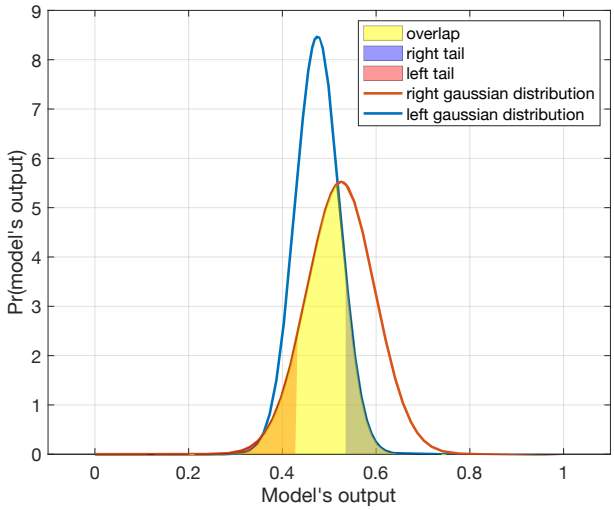

Figure III.62 *Model abstention from classification.*

III.1.3.1 Improved classification of motorcycle rear stroke suspension sensor faults: Dataset and Network

The proposed methodology was applied to increase the reliability of an IFD model for rear stroke sensors. These systems can promptly detect sensor failures so that appropriate recovery strategies or even fault accommodation strategies can be activated in a timely manner, which is critical in many automotive applications.

In terms of sensors, two linear potentiometers and a gyroscope (Table III.8) were mounted on a motorcycle to measure the vertical dynamics of the two-wheeled vehicle. The linear potentiometers used (front and rear potentiometers) are the SLS3130 Penny and Giles and the gyroscope used is a STL3GD20. Data acquisition from these sensors is done by means of a data logger equipped with a Controller Area Network (CAN) bus peripheral to interface these sensors with all electronic control units mounted on the motorcycle. These data were properly collected and used to train and evaluate the performance of the classification model (Capriglione et al., 2019; Capriglione et al., 2020).

Table III.8 *Sensors employed to measure the vertical dynamics of the motorcycle.*

Sensor Type	***Model***	***Characteristics***
Linear Displacement Sensor	Penny & Giles SLS130	Range: 0-200 [mm]
Gyroscope	ST L3GD20	Range: ±250 [°/s]

The faults that can occur in a potentiometer can be divided mainly into two categories: electrical faults and mechanical faults (Sidhu, 2012). The main electrical faults concern the breakage of the sensor connecting cable (short circuit or open circuit), while mechanical faults concern the breakage or wear of the sensor element. Given the physical characteristics of the sensor considered, open circuit and short circuit were considered regarding electrical failures, and aging and loss of calibration were considered regarding mechanical failures. The sensor output is set to the maximum possible value (full scale) when an open circuit condition occurs. In contrast, the output is set to the minimum possible value when a short circuit occurs. As for mechanical faults, when ageing occurs, from a specific value of the potentiometer position where the fault occurs, the acquired signal is characterized by the presence of intermittent peaks, the repetition frequency of which depends on the number of times the potentiometer passes through the position where the graphene fracture occurs. On the other hand, calibration loss manifests itself as a slight change in the slope of the sensor input/output curve, the magnitude of which depends on the degree of potentiometer wear.

The different operating conditions were grouped in four possible classes: the first class includes data related to normal sensor operation, the second class is related to electrical type faults (open circuit and short circuit), and classes three and four are related to sensor aging and calibration loss, respectively.

The model used is a multilayer Perceptron with three hidden layers (Figure III.63):

- Input layer: 32 neurons with ReLU activation function;
- Hidden layer: 16 neurons with ReLU activation function;
- Output layer: 4 neurons with softmax activation function.

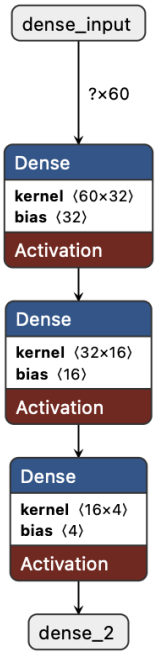

Figure III.63 *Model structure.*

This model receives as input 60 points consecutively, twenty points for three different sensors (front potentiometer, rear potentiometer, and gyroscope), of the signal of interest (scrolling through the entire acquisition window) and, based on the parameter values determined during the training phase, determines the type of failure. This information is combined with the uncertainty estimated using the law of propagation of uncertainty. Then, probabilistic inference is used to determine the confidence level associated with the assigned class.

The uncertainty information needed to apply the proposed methodology was obtained from the datasheets of the sensors used to acquire the data, assuming that the data input to the network is distributed according to a Gaussian distribution. Details on how input standard uncertainties are formed (Type A/Type B) and how input distributions are specified are provided in Section II.1.

Parallel to the structure highlighted, a network was then implemented to estimate the uncertainty on the final prediction following the principles described in the ISO-GUM on the LPU in nonlinear models; this network then propagates the uncertainty information from the input data and uses the weights determined during the training phase in the different layers to obtain this information. The only precautions to be considered during this process involve including the nonlinear terms for the softmax activation function

present in the output layer because they may represent a nonnegligible term in determining the uncertainty of the final prediction.

III.1.3.2 Experimental Results

Information on the performance of the classical network (which operates without any information on the uncertainty of the input data) in terms of the correctness of the predictions on the test set is shown in Figure III.64 in the form of a confusion matrix. This matrix thus shows the network's predictions without any information about the goodness of the prediction obtained, let alone the confidence level with which that result is given.

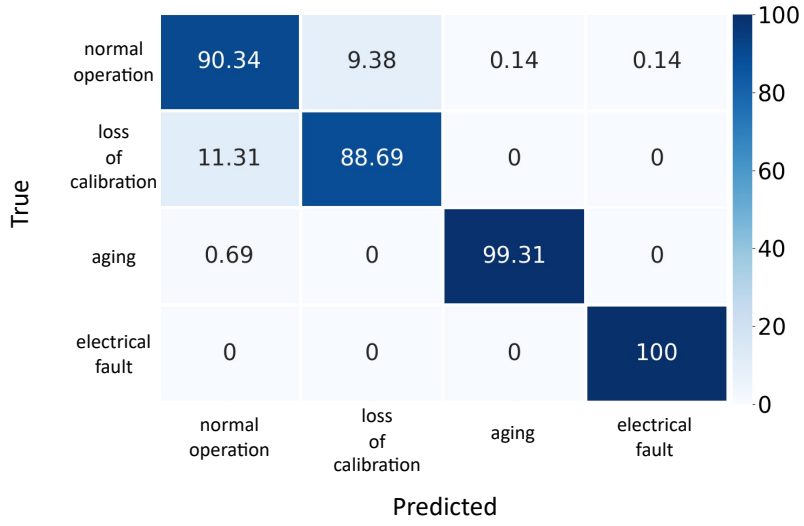

Figure III.64 *Confusion matrix.*

From the confusion matrix in Figure III.64, the baseline multi-class performance can be summarized numerically. Considering the diagonal entries as per-class recalls, the macro-averaged recall (balanced accuracy) is approximately $(90.34 + 88.69 + 99.31 + 100)/4 \approx 94.59\%$. Under equal class weighting, the corresponding macro-F1 is also approximately 94.6%, and the dominant confusions occur between normal operation and loss of calibration (i.e., the two classes most frequently exchanged by the model). This baseline quantifies the performance of standard inference when the network is forced to always output one of the four classes, without any uncertainty-aware decision rule.

The CL-based uncertainty handling adopted in the multi-class setting introduces an additional outcome, namely abstention (unclassified sample), whenever the predicted class is not sufficiently separable from its closest

competitors at the prescribed confidence level. For this reason, a standard confusion matrix (which assumes that every sample is assigned to one of the C classes) is not directly applicable once CL-thresholding is enabled, unless an explicit 'unclassified' category is introduced. In this thesis, Figures III.65-III.67 are reported primarily to provide an illustrative view of how the output distributions evolve across classes and how ambiguity emerges over time; a full test-set quantitative evaluation of CL-thresholding in the multi-class setting is outside the scope of the present experimental section.

A principled quantitative assessment of multi-class CL-thresholding should report performance jointly with coverage, i.e., the fraction of samples that receive a class label. Specifically, one should compute coverage $= \frac{N_{\text{classified}}}{N}$, and selective classification metrics (e.g., accuracy and macro-F1) computed only on the classified subset. This allows distinguishing whether improved reliability comes from genuinely better class separability or from a more conservative abstention policy. The same protocol can also be extended by adding an explicit 'unclassified' column/row to the confusion matrix to visualize how off-diagonal mass is converted into abstentions as CL increases.

It is essential to have additional information concerning the intrinsic operation of the network, considering the uncertainty of the input data to consider the possible fluctuations that might occur on the matrix due to more or less uncertain data. For this reason, combining the prediction uncertainty obtained by applying uncertainty propagation in the mathematical model of the network and statistical inference, it is possible to obtain additional information and derive conclusions about the class of fault type. Having fixed a specific value of the confidence level CL, it is possible to determine the degree of confidence with which the prediction is given.

Figures III.65, III.66 and III.67 show the results obtained for model inference on a sample set of input data (60 consecutive points-20 for each of the three sensors) by applying the proposed methodology. The classical network, without any information on the uncertainty of the input data, would assign the type of fault to be that related to an electrical fault, the latter having the highest score among those assigned to the four possible classes (only the two classes with higher scores are shown in the figures, as they are subject to possible misclassification by the network being those that might have overlap).

Accordingly, these figures are used here as a qualitative illustration of ambiguity formation across classes, whereas the quantitative multi-class evaluation of CL-thresholding is best expressed through coverage and selective macro-F1 as described above.

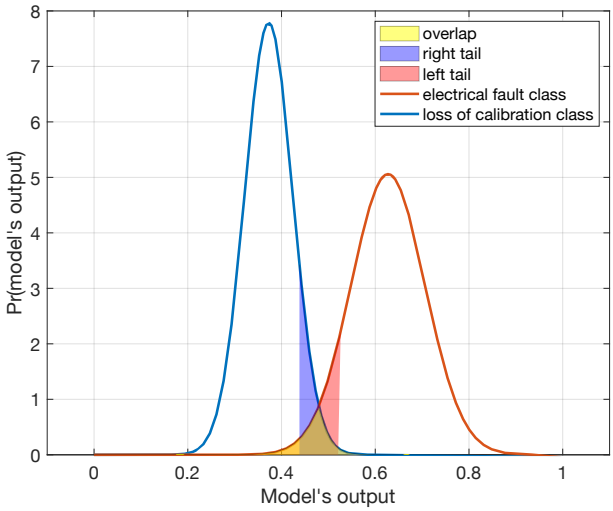

Figure III.65 *Model classification with 90% confidence level.*

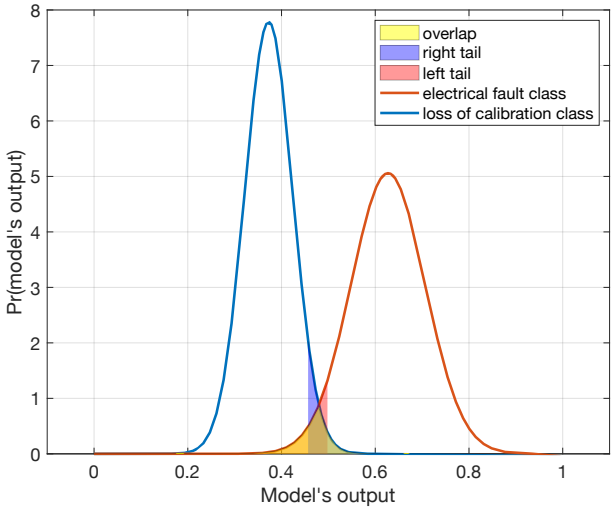

Figure III.66 *Model classification with 95% confidence level.*

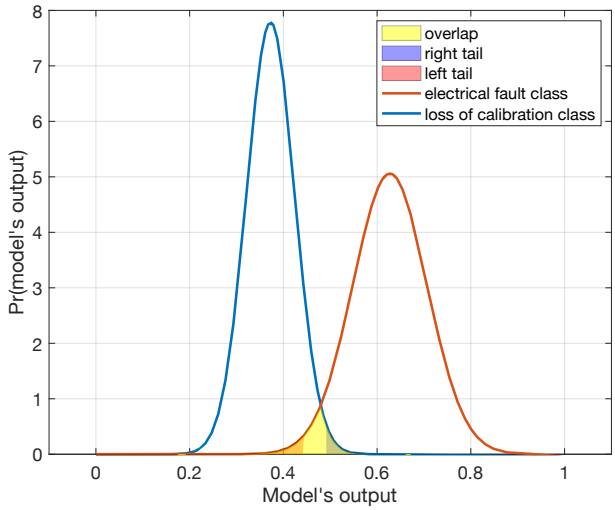

Figure III.67 *Model classification with 99% confidence level (abstention from classification).*

In Figures III.65 and III.66, a confidence level of 90% and 95%, respectively, was considered: as the tails of the two Gaussians do not entirely overlap it is possible to assign a class to the prediction, highlighting that for these confidence levels considered, the prediction of the classical model is indeed correct. Figure III.67, on the other hand, considers a 99% confidence level; in this case, however, the tails of the two Gaussians are entirely contained within the areas subtended by the opposite Gaussians, preventing the model from being exposed to the type of failure that occurred. Therefore, by increasing the confidence level and consequently the risk, samples can be classified with a higher confidence level, but, at the same time, the number of samples that are not assigned the class also increases. In contrast, reducing the risk makes it possible to classify more samples with lower confidence.

Applying these considerations to the confusion matrix, it can be said that as the confidence level changes, the number of occurrences on the matrix diagonal changes because more and more test samples can no longer be classified as the confidence level increases. This considers possible fluctuations in the input data and possible prediction errors due to the intrinsic operation of the network due to the value of the weights determined during the training phase.

Unlike the binary case discussed earlier in this chapter, where the full test-set confusion matrices were reported for each uncertainty level, the multi-class results in this subsection focus on illustrating the distributional behavior and the decision ambiguity across competing classes.

Unlike binary thresholding, multi-class confidence filtering naturally leads to a selective classification regime, where the appropriate performance summary must jointly report accuracy/F1 and coverage (fraction of non-abstained samples). In this dissertation the effect is illustrated qualitatively through the confidence plots and distribution shifts; a full test-set sweep over confidence thresholds with macro-F1-coverage curves is left as a direct extension.

III.2 Dimensionality Reduction and Signal Reconstruction under Input Uncertainty

This section does not address supervised regression in the traditional sense; instead, it focuses on uncertainty propagation in unsupervised dimensionality-reduction models, where the learning objective is signal reconstruction and the output of interest is the reconstructed signal (and its associated uncertainty).

In the context of regression, the proposed LPU methodology has been applied to an autoencoder architecture designed for the compression and subsequent decompression of a signal along with its associated uncertainty band.

The autoencoder constitutes a fundamental Artificial Neural Network architecture specifically designed to learn data representations in an unsupervised manner. Its structure, composed of an encoder and a decoder, enables information to be compressed into a reduced-dimensional latent space and subsequently reconstructed. This mechanism underpins a wide range of applications, including dimensionality reduction, data compression, noise removal, image analysis, and anomaly detection, owing to its ability to extract meaningful features from raw data while reducing computational complexity. Nevertheless, when autoencoders are employed for the compression and reconstruction of signals, the quality of the reconstruction may vary considerably, particularly in the presence of noise or anomalies.

Furthermore, autoencoders are deterministic models, which means that they always produce the same output for a given input without being able to capture the inherent uncertainty in the data, which is one of the most important assessments in metrology. To overcome these shortcomings, it is useful to supplement the classical autoencoder model with methods that effectively account for the uncertainty in the data.

To address this limitation, the use of autoencoders is extended by incorporating the proposed LPU-based methodology, which allows not only the reconstruction of the signal itself but also its associated uncertainty band (Figure III.68). In this way, the autoencoder is enhanced with the capability to provide a more accurate and reliable assessment of reconstruction quality, thereby improving its overall performance in regression tasks.

This approach enables the autoencoder not only to learn an efficient latent representation of the signal but also to accurately reconstruct the corresponding uncertainty information, thereby ensuring the consistent preservation of both the estimated value and its reliability throughout the encoding-decoding process.

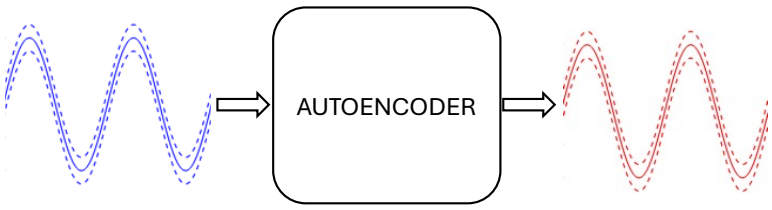

Figure III.68 *Uncertainty aware autoencoder.*

III.2.1 Uncertainty-Aware Data Reconstruction in Autoencoders

To apply the LPU to an autoencoder, it is essential to carefully analyze the mathematical relationships between the model's different layers. This involves understanding how uncertainties in the input data propagate through the encoding, bottlenecking, and decoding layers and how each transformation affects the overall uncertainty of the output.

III.2.1.1 Encoding phase

The encoding phase of an autoencoder plays a crucial role in compressing and representing the input data. In this stage, the input is passed through a sequence of layers, each applying transformations aimed at reducing dimensionality. The process starts at the input layer, which receives the raw data, and continues through successive layers where the signal is progressively compressed. The goal of this compression is to retain the most informative content while filtering out noise and redundancies. The outcome is a compact encoding that captures the most relevant features of the original data within a reduced representation space.

In convolutional autoencoders, this phase is realized using convolutional layers. Each layer applies a set of filters (kernels) that scan the input and detect important features such as edges, textures, and shapes. The convolution operation consists of computing scalar products between the filters and local regions of the input, producing activation maps that highlight the presence of learned features across different positions. As the data flows through deeper convolutional layers, the extracted features become increasingly abstract and compressed, ultimately forming a latent representation of lower dimensionality compared to the original input.

Mathematically, the convolution operation can be expressed as in equation (III.9), where x denotes the input and h the convolution kernel:

$$y[n] = \sum_{m=-\infty}^{\infty} x[m] \cdot h[n-m]. \quad \text{(III.9)}$$

III.2.1.2 Decoding phase

The decoding phase of an autoencoder is essential for decompressing the latent representation and reconstructing the original input. In this stage, the latent vector is passed through a series of layers, each applying transformations that progressively increase dimensionality. This process restores the compressed features into a signal that closely resembles the initial input. In convolutional autoencoders, decoding is typically carried out using transposed convolution:

$$y[n] = \sum_{m=-\infty}^{\infty} x[m] \cdot h[n+m]. \quad \text{(III.10)}$$

Transposed convolution, often referred to as deconvolution, can be regarded as the inverse operation of standard convolution. While conventional convolution applies filters to reduce dimensionality and produce a compressed feature map, transposed convolution expands the input by applying filters in a way that enlarges the spatial dimensions. This expansion enables the network to reconstruct a higher-dimensional output from the compact latent space representation.

Two fundamental parameters that govern convolution and transposed convolution operations are stride and padding. The stride defines the step size of the kernel as it moves across the input matrix, determining how much the kernel shifts at each position. Padding, on the other hand, refers to the addition of zeros around the borders of the input prior to convolution. By introducing padding, the network can preserve the desired output dimensions and ensure that edge regions of the input are fully included in the operation.

The interplay between stride and padding ultimately dictates the spatial dimensions of the output. For instance, appropriate padding can offset the dimensionality reduction caused by larger stride values, thereby preserving essential structural characteristics of the input. In essence, stride and padding influence not only the size of the output but also its representational quality, making them critical design choices for tailoring convolutional and transposed convolutional layers to the requirements of a given model or application.

III.2.1.3 Uncertainty Estimation

The LPU is then applied to III.9 and III.10 and the activation functions placed downstream of each convolutional and transposed convolutional layer.

Since the mathematical relationships described are simple sums and, thus, purely linear operations, the sensitivity coefficients representing the partial derivatives in can be expressed by the square of each element of the kernel matrix.

For this reason, the laws describing the measurement uncertainty at each of the convolutional and transposed convolutional layers are:

$$u_y[n] = \sqrt{\sum_{m=-\infty}^{\infty} (h[n-m])^2 \cdot (u_x[m])^2}\,, \tag{III.11}$$

$$u_y[n] = \sqrt{\sum_{m=-\infty}^{\infty} (h[n+m])^2 \cdot (u_x[m])^2}. \tag{III.12}$$

Since the operations described are all linear operations, the relationships described here for simplicity's sake for the 1D case can also be extended to 2D and 3D operations, with a simple increase in dimensionality.

The main consideration when propagating uncertainty through the network is the linearity of the activation functions applied after each layer. If the activations are not linear, the LPU cannot be applied directly. For example, when using the ReLU activation (as in this analysis), the function is linear for values greater than zero, so no additional terms need to be included in the final uncertainty estimation. In contrast, if a strongly nonlinear activation such as the sigmoid, hyperbolic tangent, or softmax is applied downstream of a convolutional layer, the direct application of the uncertainty propagation law is no longer valid. In such cases, one must either analyze the range of local linearity around the operating points or include higher-order terms in the Taylor series expansion to correctly account for the nonlinearity in the treatment.

III.2.2 Practical Use Case Scenario

The proposed methodology was applied to a practical case of the autoencoder, demonstrating its effectiveness in reconstructing the signal of interest and its uncertainty band.

This approach allows the autoencoder to not only reconstruct the primary signal but also provide a quantifiable measure of uncertainty, offering deeper insights into the confidence of the predictions. Such an improvement is essential in scenarios where understanding the variability of the data is as important as capturing its core patterns, enabling more robust and informed decision-making in fields like signal processing, risk analysis, and system diagnostics.

In this application, the autoencoder architecture shown in Figure III.69 was trained on several different frequency sine waves (including modulated signals), enabling it to capture complex patterns and nuances within the data.

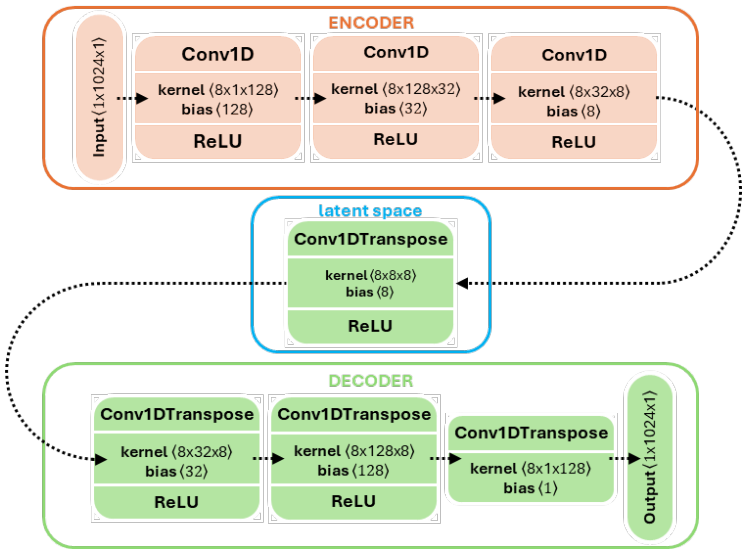

Figure III.69 *Autoencoder structure employed.*

Vectors of size 1024x1 represent the input signals used to train the network. These vectors undergo a compression process through the convolutional layers of the encoder, where convolution operations reduce dimensionality and extract the most relevant features of the original signal. During this phase, the model learns to represent the data in a more compact latent space. Next, the compressed signal passes through the transposed convolutional layers of the decoder. These layers perform decompression operations, restoring the original dimensionality of the signal. This process aims to obtain the correctly reconstructed signal, which must be as similar as possible to the input signal. Training the network continuously improves compression and decompression capability, allowing the model to learn valuable representations and preserve meaningful information during reconstruction.

The inference step on unknown signals by the autoencoder for the reconstruction of the input signal was accompanied by the propagation of the

uncertainty of the input data according to the proposed methodology. This process is crucial for assessing and representing the variability and uncertainty associated with the original measurements. During inference, the new model thus created not only reconstructs the signal but also considers uncertainties in the input data, which may arise from variables such as noise, measurement errors, or fluctuations in the original signals. The propagation of these uncertainties allows a band of uncertainty to be generated around the reconstructed signal, thus providing a more robust and realistic representation of the estimates produced by the autoencoder.

The uncertainty of the input data to the network was determined as the maximum value of 1% of the input signal.

This makes it possible to quantify the level of noise or fluctuation that may affect the system's behavior. Since data can come from different sources and have varying characteristics, adopting this percentage criterion offers a standardized measure to assess the model's reliability.

The presented Figures III.70, III.71 and III.72 show a detailed comparison between a portion of the input signal with its corresponding uncertainty band and its reconstructed counterpart accompanied by the corresponding reconstructed uncertainty band.

This visual representation highlights the network's performance in reconstructing the original signal and illustrates how the methodology adopted allows the uncertainty band associated with the reconstructed signal to be determined (Figure III.70). In some cases, the network fails to reconstruct the input signal perfectly, as can be observed by the lack of complete overlap between the reconstructed signal, with its associated uncertainty band, and the original signal (Figure III.71). On the contrary, in other cases, the network succeeds in picking up variations in the signal and reconstructing it together with its uncertainty band (Figure III.72).

In Figures III.70-III.72 the shaded bounds should be interpreted as propagated input-uncertainty envelopes obtained via local uncertainty propagation through the reconstruction model. In other words, they quantify how sensitive the reconstructed waveform $\hat{y}(n)$ is to the prescribed perturbations of the input around its nominal value. Therefore, these bounds are not meant to be a fully calibrated probabilistic ‘prediction interval’ that explains all sources of discrepancy between the original signal $y(n)$ and its reconstruction $\hat{y}(n)$. With this interpretation, it is expected that some segments (e.g., Figure III.71) may show portions of the reference signal outside the band: this indicates that the reconstruction residual in that window is driven by effects that are not captured by input-uncertainty propagation alone, such as representation/approximation limits of the latent space, training-related bias, or local out-of-distribution patterns. Conversely, Figure III.72 illustrates a window where the reconstruction residual is largely consistent with the propagated input variability, so the envelope provides a more informative uncertainty description.

If a strictly calibrated uncertainty interval for reconstruction error is required (i.e., a band that attains a prescribed empirical coverage for $y(n)$), the propagated-input envelope can be complemented with an additional term representing model/reconstruction uncertainty. Conceptually, this corresponds to combining the propagated input-uncertainty contribution with a residual uncertainty component derived from reconstruction errors on a calibration set or applying a simple post-hoc scaling of $u_{\hat{y}}(n)$ to match a target coverage level. Importantly, this does not change the propagation mechanism presented here; it only changes the interpretation from 'input-uncertainty sensitivity envelope' to 'prediction interval' by accounting for additional error sources.

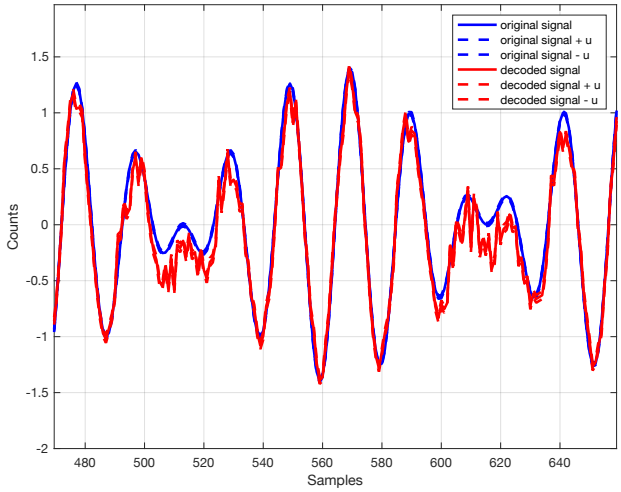

Figure III.70 *Comparison of the original signal and its uncertainty band with reconstructed counterparts.*

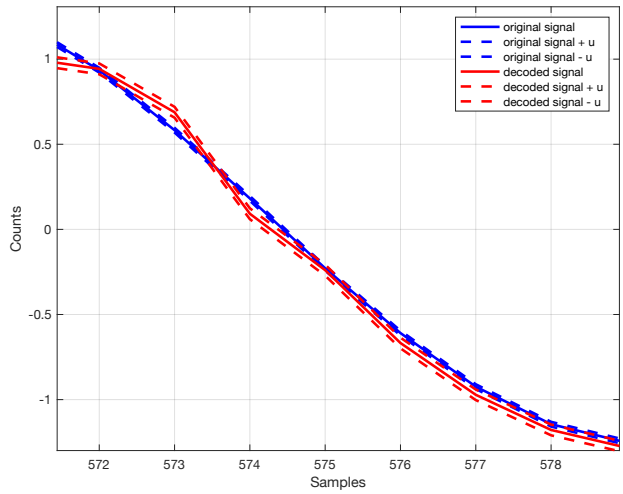

Figure III.71 *Zoom on the comparison of the original signal and its uncertainty band with reconstructed counterparts - uncorrect reconstruction.*

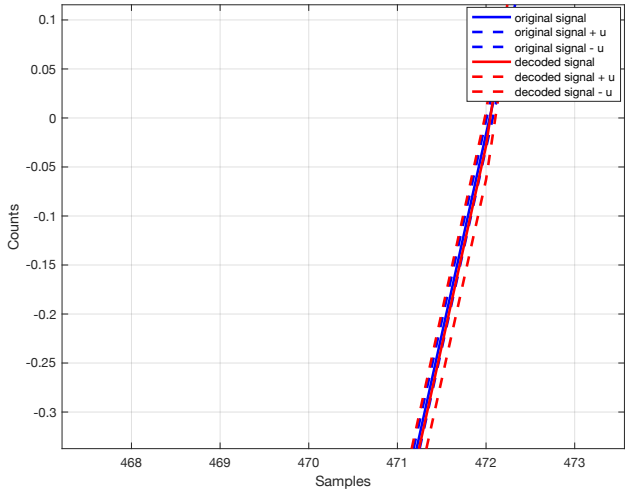

Figure III.72 *Zoom on the comparison of the original signal and its uncertainty band with reconstructed counterparts - correct reconstruction.*

From an evaluation standpoint, reconstruction uncertainty can be assessed through standard calibration diagnostics, such as the empirical coverage of the interval $\left[\hat{y}(n) - k\,u_{\hat{y}}(n), \hat{y}(n) + k\,u_{\hat{y}}(n)\right]$ and the normalized residual $|\,y(n) - \hat{y}(n)\,|/u_{\hat{y}}(n)$. In this section, the focus is on clarifying the qualitative

behaviour of propagated uncertainty in reconstruction (Figures III.70-III.72) and on illustrating how the uncertainty scale evolves with the prescribed input perturbation (Figure III.73).

Figure III.73 furthermore shows the ratio between output uncertainty and input uncertainty, highlighting how the reconstructed uncertainty is greater than the input uncertainty. Analyzing this ratio is crucial for understanding the reliability of measurements and evaluating the propagation of uncertainties in the autoencoder compression and decompression process. The graph also shows that the uncertainty ratio tends to increase significantly in areas where the input signal's slope changes quickly. This phenomenon occurs because the reconstructed signal has a more significant discrepancy from the original in precisely these regions. A less accurate reconstruction of the input can be observed in these areas due to the greater complexity of capturing the sudden changes in the signal. This leads to a lower fidelity of the overall reconstruction and, consequently, a higher level of uncertainty. Furthermore, this ratio tends to decrease at the beginning and end of the sequence, as fewer samples are used in the convolution and transposed convolution formulas in these areas than in the center of the sequence, where more is used, leading to a consequent increase in uncertainty in the reconstruction.

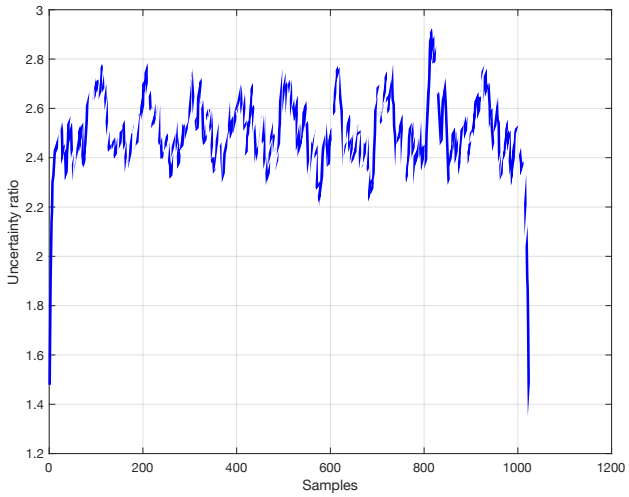

Figure III.73 *Ratio of output uncertainty to input uncertainty.*

The main advantage of applying the proposed method is that it allows obtaining the output uncertainty with a single iteration, making it significantly more efficient than the Monte Carlo method. The latter, in fact, requires many simulations to estimate the output distribution, making it highly time-consuming and demanding in terms of computational resources.

This analysis demonstrates that the proposed methodology is highly effective in reconstructing the uncertainty band associated with the variability of the input signal, successfully capturing its stochastic nature. This capability is crucial in applications where understanding the uncertainty of the signal is essential for making reliable predictions and informed decisions. To further enhance its performance, incorporating the contribution of the model coefficients h to the overall uncertainty could provide a more comprehensive uncertainty estimation. This improvement would allow the method to highlight areas where the model's predictions are less reliable, offering deeper insights into its limitations while reinforcing its strengths. Overall, the proposed methodology already demonstrates strong effectiveness in uncertainty quantification, and with further refinements, it has the potential to provide an even more robust and informative assessment of model reliability.

III.2.3 Findings and Remarks

Autoencoders are a powerful tool in applications involving dimensional reduction, data compression, and noise removal. However, the quality of reconstruction can be affected by several factors that highlight the need to incorporate techniques that allow a more accurate assessment of the reliability of the reconstructed signal. To address these challenges, a more robust approach is needed to assess the confidence and accuracy of the reconstruction process. In this dissertation, an uncertainty propagation law-based methodology aimed at improving the performance of autoencoders by incorporating measurement uncertainty directly into the reconstruction process was presented. Incorporating the uncertainty of the input signal not only enables a more complete and reliable reconstruction of the signal but also provides a measure of the quality of the reconstruction. This methodology ensures that inherent uncertainties in the input data propagate through the network, which is critical in applications where understanding the reliability of the output is as important as the output itself. On the other hand, a possible improvement is to also consider the epistemic uncertainty of the model. This would allow the analysis to integrate all contributions of uncertainty, showing greater uncertainty in areas where the model cannot accurately reconstruct the signal of interest. In conclusion, the proposed methodology represents a step forward in improving the performance of autoencoders.

Chapter IV
Conclusions and Future Works

The rapid proliferation of AI systems for classification and prediction has profoundly transformed both scientific research and industrial applications. These technologies have achieved unprecedented performance across a wide range of tasks, from image recognition and natural language processing to predictive modeling in engineering, medicine, and finance. However, the deployment of AI systems in safety-critical and high-stakes environments, such as autonomous vehicles, healthcare diagnostics, or industrial process control, raises pressing concerns regarding the reliability, robustness, and trustworthiness of their outputs. In these contexts, even small errors or unquantified uncertainties in predictions can lead to catastrophic consequences, making the rigorous assessment of uncertainty not just desirable, but essential.

The core focus of this research has been the development of a methodology for quantifying and propagating uncertainty in Artificial Neural Networks, with particular emphasis on analytical approaches grounded in the Law of Propagation of Uncertainty as formalized in the ISO-GUM. By systematically linking the known uncertainties of input data to the resulting output uncertainty, this approach allows for a more transparent and interpretable assessment of ANN predictions. In regression problems, the propagated uncertainty provides a direct measure of the confidence interval associated with the predicted values. In classification problems, this framework enables the quantification of class-level confidence, offering a probabilistic understanding of the model's predictions beyond conventional outputs.

The proposed methodology has demonstrated significant advantages. It is computationally efficient, requiring only the evaluation of derivatives rather than expensive Monte Carlo simulations, and it provides a clear analytical understanding of how input uncertainties influence outputs. When integrated with a probabilistic framework, this approach enhances the robustness, reliability, and interpretability of ANN models, offering practical benefits

across diverse domains including signal processing, control systems, medical diagnostics, and predictive maintenance. By explicitly accounting for the uncertainty in input measurements and model responses, this framework provides a pathway toward AI systems that are both trustworthy and auditable, addressing one of the key limitations in current AI deployment in safety-critical applications.

While the proposed framework provides an ISO-GUM-aligned and computationally efficient route to propagate measurement (input) uncertainty through deterministic neural networks, its current validation is intentionally scoped to settings where a local linearization around the nominal input is informative. Consequently, the quality of the propagated uncertainty depends on the degree of nonlinearity of the network in the operating region, the adequacy of the assumed input uncertainty model (e.g., Gaussian approximation and independence when adopted), and the stability of the activation operating points (e.g., proximity to saturation or switching regions). Importantly, in its present form the method quantifies aleatoric uncertainty induced by input measurements, whereas epistemic/model uncertainty is treated as a separate contribution and is not fully integrated into the uncertainty budget.

A first, concrete direction for future work is to strengthen the metrological grounding by extending the framework toward a more explicit uncertainty budget consistent with ISO-GUM practice: introducing a systematic decomposition of contributions (measurement channels, preprocessing/normalization, and model sensitivity terms), treating correlated inputs via a full input covariance matrix rather than independent components, and exploring non-Gaussian input distributions where appropriate. In the same spirit, it would be valuable to complement the local linearized propagation with standardized coverage assessments, i.e., verifying how often predicted coverage intervals contain the true values under controlled perturbations, and discussing the link between “coverage factor/confidence level” and the assumptions (linearity, normality) adopted for propagation.

A second direction is the integration of mixed uncertainty (measurement + model) within a unified framework. Practically, this can be approached by combining the proposed LPU propagation with lightweight approximations of epistemic uncertainty (e.g., ensemble- or Laplace-style approximations) and by clarifying how these components should be aggregated (e.g., independence assumptions versus joint modeling). Additional extensions include: generalizing the propagation to vector outputs with full output covariance (important for multiclass settings and structured outputs), handling architectures with non-smooth elements through principled smoothing or piecewise analyses, and providing a reusable software workflow (e.g., a MATLAB-oriented implementation guideline) that automates sensitivity extraction, verifies linear-regime validity, and documents assumptions in an auditable manner.

Beyond these immediate methodological extensions within classical computing, the emerging field of quantum computing offers exciting new opportunities for future research. Quantum systems inherently possess probabilistic behavior and can process complex, high-dimensional data in ways that classical computers cannot. The convergence of neural network uncertainty quantification with quantum computing frameworks could create a new paradigm for information processing, where both data-driven and technology-driven uncertainties are rigorously incorporated into predictive models. In such a scenario, ANNs could be deployed on quantum platforms with a principled understanding of the combined uncertainties arising from both the input data and the underlying computational substrate, paving the way for next-generation AI systems that are faster, more efficient, and uncertainty-aware by design.

This doctoral project, therefore, represents more than a technical contribution; it opens a conceptual window toward the responsible and reliable deployment of AI systems in complex, high-stakes environments. By providing a structured methodology for analytically estimating uncertainty in ANNs, it lays the groundwork for future advancements where AI models are not only performant but also robust, interpretable, and deployable in conjunction with emerging quantum technologies. In the long term, this research could contribute to the creation of a new class of AI systems capable of quantifying their own reliability, fostering trust and enabling safe integration into domains where decision-making is critical, and errors are costly. Ultimately, the integration of classical and quantum frameworks for uncertainty-aware AI represents a promising frontier in the evolution of intelligent systems, with the potential to revolutionize both the theoretical foundations and practical applications of Artificial Intelligence.

References

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., ... & Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information fusion, 76, 243-297.

Abdelaziz, A. H., Watanabe, S., Hershey, J. R., Vincent, E., & Kolossa, D. (2015, September). Uncertainty propagation through deep neural networks. In Interspeech 2015.

Abdullah, A. A., Hassan, M. M., & Mustafa, Y. T. (2024). Leveraging Bayesian deep learning and ensemble methods for uncertainty quantification in image classification: A ranking-based approach. Heliyon, 10(2).

Afaq, S., & Rao, S. (2020). Significance of epochs on training a neural network. Int. J. Sci. Technol. Res, 9(06), 485-488.

Alpaydin, E. (2021). Machine learning. MIT press.

Arbib, M. A. (Ed.). (2003). The handbook of brain theory and neural networks. MIT press.

Astudillo, R. F., & da Silva Neto, J. P. (2011, October). Propagation of Uncertainty Through Multilayer Perceptrons for Robust Automatic Speech Recognition. In Interspeech (pp. 461-464).

Ayhan, M. S., & Berens, P. (2018, July). Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. In Medical Imaging with Deep Learning.

Bachstein, S. (2019). Uncertainty quantification in deep learning. Master Thesis.

Banos, O., Garcia, R., Holgado-Terriza, J. A., Damas, M., Pomares, H., Rojas, I., ... & Villalonga, C. (2014, December). mHealthDroid: a novel framework for agile development of mobile health applications. In International workshop on ambient assisted living (pp. 91-98). Cham: Springer International Publishing.

Basheer, I. A., & Hajmeer, M. (2000). Artificial neural networks: fundamentals, computing, design, and application. Journal of microbiological methods, 43(1), 3-31.

Bebis, G., & Georgiopoulos, M. (2002). Feed-forward neural networks. Ieee Potentials, 13(4), 27-31.

BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML. Evaluation of measurement data — Guide to the expression of uncertainty in measurement. Joint Committee for Guides in Metrology, JCGM 100:2008. doi:10.59161/JCGM100-2008E.

BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML. Evaluation of measurement data — Supplement 1 to the "Guide to the expression of uncertainty in measurement" — Propagation of distributions using a Monte Carlo method. Joint Committee for Guides in Metrology, JCGM 101:2008. doi:10.59161/JCGM101-2008.

BIPM, IEC, IFCC, ILAC, ISO, IUPAC, IUPAP, and OIML. (2012). International vocabulary of metrology — Basic and general concepts and associated terms (VIM). Joint Committee for Guides in Metrology, JCGM 200:2012. doi:10.59161/JCGM200-2012.

Bre, F., Gimenez, J. M., & Fachinotti, V. D. (2018). Prediction of wind pressure coefficients on building surfaces using artificial neural networks. Energy and Buildings, 158, 1429-1441.

Brownlee, J. (2021). Ensemble learning algorithms with Python: Make better predictions with bagging, boosting, and stacking. Machine Learning Mastery.

Bühlmann, P. (2011). Bagging, boosting and ensemble methods. In Handbook of computational statistics: Concepts and methods (pp. 985-1022). Berlin, Heidelberg: Springer Berlin Heidelberg.

Burns, A., Greene, B. R., McGrath, M. J., O'Shea, T. J., Kuris, B., Ayer, S. M., ... & Cionca, V. (2010). SHIMMER™–A wireless sensor platform for noninvasive biomedical research. IEEE Sensors Journal, 10(9), 1527-1534.

Calin, O. (2020). Deep learning architectures. New York City: Springer International Publishing.

Capriglione, D., Carratu, M., Pietrosanto, A., & Sommella, P. (2019). Online fault detection of rear stroke suspension sensor in motorcycle. IEEE Transactions on Instrumentation and Measurement, 68(5), 1362-1372.

Capriglione, D., Carratù, M., Pietrosanto, A., & Sommella, P. (2020). Soft sensors for instrument fault accommodation in semiactive motorcycle suspension systems. IEEE Transactions on Instrumentation and Measurement, 69(5), 2367-2376.

Caruana, R., Lawrence, S., & Giles, C. (2000). Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. Advances in neural information processing systems, 13.

Cilimkovic, M. (2015). Neural networks and back propagation algorithm. Institute of Technology Blanchardstown, Blanchardstown Road North Dublin, 15(1), 18.

Daruna, A., Gong, Y., Rajvanshi, A., Chiu, H. P., & Yao, Y. (2023). Uncertainty Propagation through Trained Deep Neural Networks Using Factor Graphs. arXiv preprint arXiv:2312.05946

Diamzon, J., & Venturi, D. (2025). Uncertainty propagation in feed-forward neural network models. arXiv preprint arXiv:2503.21059.

Dogo, E. M., Afolabi, O. J., Nwulu, N. I., Twala, B., & Aigbavboa, C. O. (2018, December). A comparative analysis of gradient descent-based optimization algorithms on convolutional neural networks. In 2018 international conference on computational techniques, electronics and mechanical systems (CTEMS) (pp. 92-99). IEEE.

Fergus, P., & Chalmers, C. (2022). Applied deep learning. Computational intelligence methods and applications.

Gal, Y., & Ghahramani, Z. (2015). Bayesian convolutional neural networks with Bernoulli approximate variational inference. arXiv preprint arXiv:1506.02158.

Gal, Y., & Ghahramani, Z. (2016, June). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In international conference on machine learning (pp. 1050-1059). PMLR.

Gast, J., & Roth, S. (2018). Lightweight probabilistic deep networks. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 3369-3378).

Gawlikowski, J., Tassi, C. R. N., Ali, M., Lee, J., Humt, M., Feng, J., ... & Zhu, X. X. (2023). A survey of uncertainty in deep neural networks. Artificial Intelligence Review, 56(Suppl 1), 1513-1589.

Ghosh, J., & Nag, A. (2001). An overview of radial basis function networks. Radial basis function networks 2: new advances in design, 1-36.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2). Cambridge: MIT press.

Gupta, R. (2020, February). A survey on machine learning approaches and its techniques. In 2020 IEEE International Students' Conference on Electrical, Electronics and Computer Science (SCEECS) (pp. 1-6). IEEE.

Han, P. C. (2026). Artificial Intelligence in Academic Research: Applications, Challenges, and Future Directions. In Foundations and Frameworks for AI in Education (pp. 451-476). IGI Global Scientific Publishing.

Hansen, L. K., & Salamon, P. (2002). Neural network ensembles. IEEE transactions on pattern analysis and machine intelligence, 12(10), 993-1001.

Haykin, S. (2009). Neural networks and learning machines, 3/E. Pearson Education India.

Hebb, D. O. (1949). The organization of behavior: A neuropsychological theory. (Reprinted 2005, Psychology Press.).

Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., ... & Kingsbury, B. (2012). Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. IEEE Signal processing magazine, 29(6), 82-97.

Hramov, A. E., Frolov, N. S., Maksimenko, V. A., Makarov, V. V., Koronovskii, A. A., Garcia-Prieto, J., ... & Pisarchik, A. N. (2018). Artificial neural network detects human uncertainty. Chaos: An Interdisciplinary Journal of Nonlinear Science, 28(3).

https://netron.app

https://www.kaggle.com/datasets/abhinand05/magic-gamma-telescope-dataset

https://stepup.ai/test_time_data_augmentation/

https://www.kaggle.com/datasets/mssmartypants/rice-type-classification

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. Machine learning, 110(3), 457-506.

Igiri, C. P., Anyama, O. U., & Silas, A. I. (2015). Effect of learning rate on artificial neural network in machine learning.

Islam, M., Chen, G., & Jin, S. (2019). An overview of neural network. American Journal of Neural Networks and Applications, 5(1), 7-11.

Janocha, K., & Czarnecki, W. M. (2017). On loss functions for deep neural networks in classification. arXiv preprint arXiv:1702.05659.

Jeon, W., Ko, G., Lee, J., Lee, H., Ha, D., & Ro, W. W. (2021). Deep learning with GPUs. In Advances in computers (Vol. 122, pp. 167-215). Elsevier.

Jiang, Y., Li, X., Luo, H., Yin, S., & Kaynak, O. (2022). Quo vadis artificial intelligence?. Discover Artificial Intelligence, 2(1), 4.

Kabir, H. D., Khosravi, A., Hosen, M. A., & Nahavandi, S. (2018). Neural network-based uncertainty quantification: A survey of methodologies and applications. IEEE access, 6, 36218-36234.

Kanal, L. N., & Lemmer, J. F. (Eds.). (2014). Uncertainty in artificial intelligence (Vol. 4). Elsevier.

Kasiviswanathan, K. S., Sudheer, K. P., & He, J. (2016). Quantification of prediction uncertainty in artificial neural network models. In Artificial neural network modelling (pp. 145-159). Cham: Springer International Publishing.

Kendall, A., & Gal, Y. (2017). What uncertainties do we need in bayesian deep learning for computer vision?. Advances in neural information processing systems, 30.

Khashei, M., Hamadani, A. Z., & Bijari, M. (2012). A novel hybrid classification model of artificial neural networks and multiple linear regression models. Expert Systems with Applications, 39(3), 2606-2620.

Kingsford, C., & Salzberg, S. L. (2008). What are decision trees?. Nature biotechnology, 26(9), 1011-1013.

Kruse, R., Mostaghim, S., Borgelt, C., Braune, C., & Steinbrecher, M. (2022). Multi-layer perceptrons. In Computational intelligence: a methodological introduction (pp. 53-124). Cham: Springer International Publishing.

Kufel, J., Bargieł-Łączek, K., Kocot, S., Koźlik, M., Bartnikowska, W., Janik, M., ... & Gruszczyńska, K. (2023). What is machine learning, artificial neural networks and deep learning?—Examples of practical applications in medicine. Diagnostics, 13(15), 2582.

Kukreja, H., Bharath, N., Siddesh, C. S., & Kuldeep, S. (2016). An introduction to artificial neural network. Int J Adv Res Innov Ideas Educ, 1(5), 27-30.

Lakshminarayanan, B., Pritzel, A., & Blundell, C. (2017). Simple and scalable predictive uncertainty estimation using deep ensembles. Advances in neural information processing systems, 30.

Li, Y., & Gal, Y. (2017, July). Dropout inference in bayesian neural networks with alpha-divergences. In International conference on machine learning (pp. 2052-2061). PMLR.

Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J. (2021). A survey of convolutional neural networks: analysis, applications, and prospects. IEEE transactions on neural networks and learning systems, 33(12), 6999-7019.

Liu, J., Shen, Z., He, Y., Zhang, X., Xu, R., Yu, H., & Cui, P. (2021). Towards out-of-distribution generalization: A survey. arXiv preprint arXiv:2108.13624.

Ludwig, B., Gruber, M., Jung, B., & Eichstädt, S. (2025). Application of uncertainty propagation in deep neural networks: Innovations in metrology and the introduction of QuadLU activation function. Measurement: Sensors, 101790.

Malmström, M. (2023). Approximative Uncertainty in Neural Network Predictions. Linkopings Universitet (Sweden).

Mammone, A., Turchi, M., & Cristianini, N. (2009). Support vector machines. Wiley Interdisciplinary Reviews: Computational Statistics, 1(3), 283-289.

Mara, T. A., Delay, F., Lehmann, F., & Younes, A. (2016). A comparison of two Bayesian approaches for uncertainty quantification. Environmental Modelling & Software, 82, 21-30.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. The bulletin of mathematical biophysics, 5(4), 115-133.

Medsker, L. R., & Jain, L. (2001). Recurrent neural networks. Design and applications, 5(64-67), 2.

Mercioni, M. A., & Holban, S. (2020, May). The most used activation functions: Classic versus current. In 2020 International Conference on Development and Application Systems (DAS) (pp. 141-145). IEEE.

Milanés-Hermosilla, D., Trujillo Codorniú, R., López-Baracaldo, R., Sagaró-Zamora, R., Delisle-Rodriguez, D., Villarejo-Mayor, J. J., & Nunez-Alvarez, J. R. (2021). Monte carlo dropout for uncertainty estimation and motor imagery classification. Sensors, 21(21), 7241.

Minsky, M., & Papert, S. (1969). Perceptrons cambridge. MA: MIT Press. zbMATH.

Monchot, P., Coquelin, L., Petit, S. J., Marmin, S., Le Pennec, E., & Fischer, N. (2023). Input uncertainty propagation through trained neural networks. In International Conference on Machine Learning 2023.

Montesinos López, O. A., Montesinos López, A., & Crossa, J. (2022). Fundamentals of artificial neural networks and deep learning. In Multivariate statistical machine learning methods for genomic prediction (pp. 379-425). Cham: Springer International Publishing.

Namatēvs, I. (2017). Deep convolutional neural networks: Structure, feature extraction and training. Information Technology and Management Science, 20(1), 40-47.

Nguyen, V. L., Destercke, S., & Hüllermeier, E. (2019, October). Epistemic uncertainty sampling. In International Conference on Discovery Science (pp. 72-86). Cham: Springer International Publishing.

O'shea, K., & Nash, R. (2015). An introduction to convolutional neural networks. arXiv preprint arXiv:1511.08458.

Odegua, R. (2019, March). An empirical study of ensemble techniques (bagging, boosting and stacking). In Proc. Conf.: Deep Learn. IndabaXAt.

Paullada, A., Raji, I. D., Bender, E. M., Denton, E., & Hanna, A. (2021). Data and its (dis) contents: A survey of dataset development and use in machine learning research. Patterns, 2(11).

Plevris, V., Solorzano, G., Bakas, N. P., & Ben Seghier, M. E. A. (2022). Investigation of performance metrics in regression analysis and machine learning-based prediction models.

Psaros, A. F., Meng, X., Zou, Z., Guo, L., & Karniadakis, G. E. (2023). Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. Journal of Computational Physics, 477, 111902.

Qamar, R., & Zardari, B. A. (2023). Artificial neural networks: An overview. Mesopotamian Journal of Computer Science, 2023, 124-133.

Rasamoelina, A. D., Adjailia, F., & Sinčák, P. (2020, January). A review of activation function for artificial neural network. In 2020 IEEE 18th world symposium on applied machine intelligence and informatics (SAMI) (pp. 281-286). IEEE.

Rodríguez, A. I., & Buitrago, X. D. (2022). How to choose an activation function for deep learning. Tekhnê, 19(1), 23-32.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review, 65(6), 386.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. nature, 323(6088), 533-536.

Santos, C. F. G. D., & Papa, J. P. (2022). Avoiding overfitting: A survey on regularization methods for convolutional neural networks. ACM Computing Surveys (Csur), 54(10s), 1-25.

Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. Towards Data Sci, 6(12), 310-316.

Sharma, V., Rai, S., & Dev, A. (2012). A comprehensive study of artificial neural networks. International Journal of Advanced research in computer science and software engineering, 2(10).

Shirmohammadi, S., & Al Osman, H. (2021). Machine learning in measurement part 1: Error contribution and terminology confusion. IEEE Instrumentation & Measurement Magazine, 24(2), 84-92.

Shirmohammadi, S., Amiri, M. H., & Al Osman, H. (2024). Measurement Methodology. IEEE Instrumentation & Measurement Magazine, 27(7), 37-45.

Sidhu, K. (2012). Understanding Linear Position Sensing Technologies.

Shirmohammadi, S., Wang, F., & Hsu, C. H. (2025). Review and performance evaluation of uncertainty quantification in data-driven AI-assisted measurements. IEEE Open Journal of Instrumentation and Measurement.

Simon, P. (2013). Too big to ignore: the business case for big data (Vol. 72). John Wiley & Sons.

Soize, C. (2017). Uncertainty quantification (Vol. 23). Springer International Publishing AG.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

Sutton, R. S., & Barto, A. G. (1998). Reinforcement learning: An introduction (Vol. 1, No. 1, pp. 9-11). Cambridge: MIT press.

Taye, M. M. (2023). Understanding of machine learning with deep learning: architectures, workflow, applications and future directions. Computers, 12(5), 91.

Titensky, J. S., Jananthan, H., & Kepner, J. (2018, October). Uncertainty propagation in deep neural networks using extended kalman filtering. In 2018 IEEE MIT Undergraduate Research Technology Conference (URTC) (pp. 1-4). IEEE.

Toosi, A., Bottino, A. G., Saboury, B., Siegel, E., & Rahmim, A. (2021). A brief history of AI: how to prevent another winter (a critical review). PET clinics, 16(4), 449-469.

Tripathy, R. K., & Bilionis, I. (2018). Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification. Journal of computational physics, 375, 565-588.

Tyralis, H., & Papacharalampous, G. (2024). A review of predictive uncertainty estimation with machine learning. Artificial Intelligence Review, 57(4), 94.

Van Amersfoort, J., Smith, L., Teh, Y. W., & Gal, Y. (2020, November). Uncertainty estimation using a single deep deterministic neural network. In International conference on machine learning (pp. 9690-9700). PMLR.

Van Engelen, J. E., & Hoos, H. H. (2020). A survey on semi-supervised learning. Machine learning, 109(2), 373-440.

Van Katwyk, P., Fox-Kemper, B., Seroussi, H., Nowicki, S., & Bergen, K. J. (2023). A variational LSTM emulator of sea level contribution from the Antarctic ice sheet. Journal of Advances in Modeling Earth Systems, 15(12), e2023MS003899.

Verbraeken, J., Wolting, M., Katzy, J., Kloppenburg, J., Verbelen, T., & Rellermeyer, J. S. (2020). A survey on distributed machine learning. Acm computing surveys (csur), 53(2), 1-33.

Wang, G., Li, W., Aertsen, M., Deprest, J., Ourselin, S., & Vercauteren, T. (2018). Test-time augmentation with uncertainty estimation for deep learning-based medical image segmentation.

Wang, T., Wang, Y., Zhou, J., Peng, B., Song, X., Zhang, C., ... & Yan, L. K. (2025). From aleatoric to epistemic: Exploring uncertainty quantification techniques in artificial intelligence. arXiv preprint arXiv:2501.03282.

Weiss, K., Khoshgoftaar, T. M., & Wang, D. (2016). A survey of transfer learning. Journal of Big data, 3(1), 9.

Wu, Y. C., & Feng, J. W. (2018). Development and application of artificial neural network. Wireless Personal Communications, 102(2), 1645-1656.

Yang, C. I., & Li, Y. P. (2023). Explainable uncertainty quantifications for deep learning-based molecular property prediction. Journal of Cheminformatics, 15(1), 13.

Yegnanarayana, B. (2009). Artificial neural networks. PHI Learning Pvt. Ltd..

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of CNN and RNN for natural language processing. arXiv preprint arXiv:1702.01923.

Ying, X. (2019, February). An overview of overfitting and its solutions. In Journal of physics: Conference series (Vol. 1168, p. 022022). IOP Publishing.

Zhang, J., Yin, J., & Wang, R. (2020). Basic framework and main methods of uncertainty quantification. Mathematical Problems in Engineering, 2020(1), 6068203.

Zhang, S., Singh, M., Menolascino, D., & Ching, S. (2025). Estimating uncertainty from feed-forward network based sensing using quasi-linear approximation. Neural Networks, 188, 107376.

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., ... & Wen, J. R. (2023). A survey of large language models. arXiv preprint arXiv:2303.18223, 1(2).

Zhou, Z. H. (2021). Machine learning. Springer nature.